Probabilistic Inference of Spatio-Temporal Dynamics in State-Space Models

Masterarbeit

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung eines

Masterabschlusses

(M.Sc. Machine Learning)

---

vorgelegt von

Jonathan Schmidt

Eberhard Karls Universität Tübingen

Dezember 2021

Tag der mündlichen Qualifikation:	21.12.2021
Erstgutachter:	Prof. Dr. Philipp Hennig
Zweitgutachter:	Prof. Dr. Todd Ehlers
Betreuer während der Thesis:	Nicholas Krämer

Abstract

Mechanistic models with differential equations are a key component of scientific applications of machine learning. Recent work in probabilistic numerics has developed a new class of solvers for ordinary differential equations (ODEs) that phrase the solution process directly in terms of Bayesian filtering. This work continues this path in three different directions. Firstly, we show that this allows such methods to be combined very directly, with conceptual and numerical ease, with latent force models in the ODE itself. It then becomes possible to perform approximate Bayesian inference on the latent force as well as the ODE solution in a single, linear complexity pass of an extended Kalman filter / smoother – that is, at the cost of computing a single ODE solution. Secondly, to extend the probabilistic treatment to systems that also model spatial interactions, this work develops a class of probabilistic algorithms for the numerical solution of nonlinear, time-dependent partial differential equations (PDEs). Current state-of-the-art PDE solvers treat the space- and time-dimensions separately, serially, and with black-box algorithms, which obscures the interactions between spatial and temporal approximation errors and misguides the quantification of the *overall* error. To fix this issue, we introduce a probabilistic version of a technique called *method of lines*. The proposed algorithm begins with a Gaussian process interpretation of finite difference methods, which then interacts naturally with filtering-based probabilistic ODE solvers. Finally, tackling the computational efficiency of these methods, we explain the mathematical assumptions and detailed implementation schemes behind solving high-dimensional ODEs with a probabilistic numerical algorithm. This has not been possible before due to matrix-matrix operations in each solver step but is crucial for scientifically relevant problems – most importantly, the solution of discretized PDEs. All presented methods are evaluated on a range of test problems.

Kurzfassung

Mechanistische Modelle basierend auf Differentialgleichungen sind wesentlicher Bestandteil wissenschaftlicher Anwendungen maschinellen Lernens. Jüngste Arbeiten in probabilistischer Numerik haben eine neue Klasse aus Lösungsverfahren für gewöhnliche Differentialgleichungen (engl. *ordinary differential equations*, ODEs) entwickelt, welche den Lösungsprozess direkt als Bayessches Filter-Problem formulieren. Die vorliegende Arbeit setzt diesen Weg in drei verschiedene Richtungen fort. Erstens zeigen wir, dass sich solche Methoden sehr direkt, mit konzeptioneller und numerischer Leichtigkeit, mit Modellen von latenten Kräften in der ODE selbst kombinieren lassen. Dies ermöglicht approximative Bayessche Inferenz und die Lösung der ODE in einem einzigen Durchgang eines erweiterten Kalman Filters / Smoothers mit linearer Komplexität – das heißt, zu den Kosten der Berechnung einer einzigen ODE-Lösung. Zweitens wird in dieser Arbeit eine Klasse probabilistischer Algorithmen für die numerische Lösung nichtlinearer, zeitabhängiger partieller Differentialgleichungen (engl. *partial differential equation*, PDE) entwickelt, um die probabilistischen Verfahren auf Systeme auszuweiten, die auch räumliche Interaktionen modellieren. Moderne Lösungsverfahren für PDEs behandeln die räumliche und zeitliche Dimension separat, seriell und mit undurchsichtigen Algorithmen, was die Interaktionen zwischen räumlichen und zeitlichen Approximationen des Fehlers verschleiert und die Quantifizierung des *insgesamten* Fehlers irreleiten. Um dieses Problem zu beheben, stellen wir eine probabilistische Version der sogenannten *Linienmethode* (engl. *method of lines*, MOL) vor. Der vorgeschlagene Algorithmus beginnt mit einer Interpretation der Finiten-Differenzen-Methode als einen Gaußschen Prozess, welcher dann auf natürliche Weise mit Filter-basierten probabilistischen Lösungsverfahren für ODEs interagiert. Abschliessend, um sich der rechnerischen Effizienz dieser Methoden anzunehmen, erläutern wir die mathematischen Annahmen und detaillierten Implementierungsschemata hinter Lösungsverfahren für hochdimensionale ODEs mit einem probabilistisch-numerischen Algorithmus. Dies war zuvor aufgrund von Matrix-Matrix-Operationen in jedem Schritt des Lösungsverfahrens nicht möglich, ist jedoch unentbehrlich für wissenschaftlich relevante Probleme – allen voran die Lösung von diskretisierten PDEs. Alle vorgestellten Methoden werden auf einer Reihe von Testproblemen evaluiert.

Acknowledgements

I am deeply grateful for the fruitful and highly instructive collaboration with Nicholas Krämer and Nathanael Bosch and I want to extend my gratitude to Philipp Hennig for always encouraging and supporting me. I further want to thank Marvin Pförtner, Katharina Ott, Jonathan Wenger, and other members of the "Methods of Machine Learning"-group for helpful discussions and feedback, and Franziska Weiler for her patience and support in administrative questions. The entirety of this work was compiled in times of a global pandemic, and thus I want to give special thanks to Samuel Wunderlich, Florian Martin, Betty Eichler, Laura Kosik, Nina Effenberger, and Christian Fröhlich for their support in difficult times.

Note on Collaborative Work

This thesis is based on three publications:

1. Jonathan Schmidt, Nicholas Krämer, and Philipp Hennig. A probabilistic state space model for joint inference from differential equation and data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

2. Nicholas Krämer, Jonathan Schmidt, and Philipp Hennig. Probabilistic numerical method of lines for time-dependent partial differential equations. *arXiv preprint arXiv:2110.11847*

3. Nicholas Krämer*[1], Nathanael Bosch*, Jonathan Schmidt*, and Philipp Hennig. Probabilistic ODE solutions in millions of dimensions. *arXiv preprint arXiv:2110.11812*

The publications/submissions, including implementations and written manuscripts, have emerged from the work on this thesis in collaboration with Nicholas Krämer and Nathanael Bosch. 2. and 3. are currently under review for the AISTATS 2022 conference.

[1] $*$ indicates primary authorship.

Contents

1 Introduction

When developing machine learning methods for real-world problems, the question of how to integrate prior knowledge into a model remains a crucial one. Practitioners are often faced with a trade-off. On the one hand, the availability of data has been increasing tremendously due to technical advances in the storage and acquisition of data. This allows for many data-driven approaches to emerge and techniques to estimate unknown quantities from observations continue to be developed and refined. On the other hand, domain experts have to be equipped with a set of tools that allow them to incorporate their knowledge meaningfully into the system. Since long before the advent of machine learning, the natural sciences have been describing physical phenomena using differential equations. Knowledge about a mechanistic system was cast into a set of equations, which then could be analyzed. In recent years – especially with the continually increasing availability of computational resources – these mechanistic systems have gained a massive surge in popularity, especially in the context of machine learning (e.g. (Raissi et al., 2019; Lu et al., 2021; Li et al., 2021; Chen et al., 2018; Gelbrecht et al., 2021)).

With increasing complexity of the mathematical descriptions of physical phenomena, it quickly becomes tedious to analyze these systems. Therefore, numerical algorithms are commonly used to compute the unknown state over time. These methods, however, come with certain caveats. The more accurate a solution has to be, the more compute power has to be invested to obtain that solution. However, especially when trying to save some of that computational effort, numerical algorithms will always entail an unavoidable numerical error. In the absence of analytical solutions, we acknowledge the fact that the obtained approximations differ from the truth to some degree. At the same time, the fact that this error can and should be quantified and provided to practitioners that rely on numerical methods is often neglected.

This work revolves around *probabilistic numerical* (PN) (Hennig et al., 2015; Cockayne et al., 2019; Wenger et al., 2021) algorithms for solving differential equations. In particular, a central concept are probabilistic solvers that are based on Gaussian filtering (Schober et al., 2019a; Tronarp et al., 2019; Kersting et al., 2020b; Krämer and Hennig, 2020, and more). These algorithms phrase forward simulations of ordinary differential equations (ODEs) as state estimation in probabilistic state-space models, allowing for quantification of the numerical error arising from the numerical approximation of the solution. Section 2.4 will cover the concept in more detail. This work will build on this concept in mainly three different directions.

1.1 Outline

The main part of this manuscript consists of three chapters that each present recent publications that arose from the work on the present thesis. All of these build on filtering-based probabilistic solvers for differential equations in different ways and all aim to take steps towards making probabilistic inference in the context of dynamical systems more practicable.

Combining multiple sources of information in state-space models Chapter 3 presents Schmidt et al. (2021), which makes use of the fact that filtering-based solvers phrase forward simulations as probabilistic inference. The state-space model is extended by additionally incorporating empirical knowledge into the state estimation process. To mitigate between contradicting sources of information, a latent force is introduced that acts on the vector field as a time-varying parameter process, adapting the mechanistic system to fit the observed data. From another perspective, mechanistic knowledge in the form of differential equations can guide an otherwise data-driven, non-parametric model, especially in the absence of data, and correct predictions, where data cannot be trusted as much. All in all, an algorithm is presented, that solves an ODE while at the same time solving an inverse problem over a latent parameter process in linear complexity in the number of time steps.

Well-calibrated probabilistic solvers for time-dependent PDEs When considering not only temporal but also spatial diffusion in a dynamical system, one has to consider partial differential equations (PDEs). There exist methods to rewrite PDEs as ODEs by discretizing the involved differential operator that describes spatial interactions. These equations can then be treated by traditional or probabilistic numerical ODE solvers. In order to provide a PDE solution that meaningfully quantifies numerical error, it is important to not neglect the discretization of the differential operator. Otherwise, solutions quickly become overconfident. Chapter 4 presents Krämer et al. (2021b), which allows for tracking both the temporal *and* the spatial discretization error, when solving PDEs with probabilistic ODE solvers.

Are probabilistic ODE solvers practicable in very high dimensions? Especially regarding the previous paragraph, it is important to raise the issue of computational efficiency. Even though the probabilistic ODE simulation can be computed at a linear cost in the number of time steps, it remains cubic in the number of dimensions. When discretizing a PDE on a large grid, the dimension of the state increases drastically very quickly. This final main chapter, Chapter 5, presents Krämer et al. (2021a). It shows how appropriately encoded independence assumptions can speed up probabilistic ODE solutions significantly. Different levels of simplifying assumptions lead to differently efficient and stable algorithms, culminating in a formulation of a popular explicit

probabilistic ODE solver that is also linear in the dimension of the state-space. This renders the computation of ODE solutions in millions of dimensions possible, which is necessary when treating discretized PDEs.

Notation

If not otherwise stated, this work will use lowercase letters $(a, k, \dots)$ for scalars, lowercase boldface letters $(\boldsymbol{x}, \boldsymbol{v}, \dots)$ for vectors, and uppercase, boldface letters $(\boldsymbol{A}, \boldsymbol{C}, \dots)$ for matrices. For random variables, we use the same convention but with upright letters (e.g. $\mathbf{x}$ for vector-valued random variables).

2 Foundations

This chapter gives an overview of relevant foundational topics that are used throughout this work. Chapters 3 to 5 propose methods that are mostly based on differential equations models, the solution of which is regarded as a probabilistic numerical inference problem. Therefore, we introduce differential equations in Section 2.1 and discuss inference in probabilistic state-space models in Section 2.2, in order to build upon these foundations when deriving new methods. After briefly elaborating on linear systems acting on stochastic processes in Section 2.3, Section 2.4 then introduces filtering-based ODE solvers. Finally, Section 2.5 establishes basic concepts and notation for partial differential equations. The respective foundational topics are vast fields of research on their own and we will keep the overview necessarily compact here. Sections 2.1 and 2.3 are largely based on Särkkä and Solin (2019), whereas Section 2.2 closely follows Särkkä (2013). Section 2.4 is taken from Krämer et al. (2021a) and Section 2.5 is adapted from the introductory section in Krämer et al. (2021b).

2.1 Ordinary Differential Equations

Formulating mechanistic knowledge in the form of differential equations has been common practice in the natural sciences for a long time. In many applications – e.g. basic physical systems like, for instance, a pendulum, planetary motion, the development of populations of interacting species, the spread of infectious diseases, to name only a few – one can specify the rate of change of system components that interact with each other, but the actual quantities are unknown. Solving the differential equation for the unknown function then possibly yields a concrete trajectory of the system dynamics. To make these models more generally applicable and more descriptive, the equations are often parametrized and the parameters can usually be assigned concrete interpretations (like e.g. the length of the rod of a pendulum or the reproduction rate of a species).

This section will give a brief introduction to ordinary differential equations. We will begin by defining some basic principles and briefly touch on challenges that arise when solving more complex systems. Sometimes, we will employ different notations for the derivative of a function. In general, $f^{(i)}(x) \equiv \frac{\mathrm{d}^i f(x)}{\mathrm{d}x^i}$ and for lower-order derivatives we will sometimes write $\dot{x} \equiv \frac{\mathrm{d}f(x)}{\mathrm{d}x}, \ddot{x} \equiv \frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}$, and so forth.

2.1.1 Definitions

An ordinary differential equation (ODE) is an equation of the form

$$x^{(\nu)}(t) = f(t, x(t), x^{(1)}(t), \ldots, x^{(\nu-1)(t)}), \tag{2.1}$$

where the function $x(t)$ is the unknown quantity to solve for, i.e. the *solution* of the ODE. The function f is called the *vector field* and describes the dynamics of the system that the ODE defines. The ODE in Equation (2.1) is of *order* ν, since ν is the highest-order derivative that occurs in the equation. It is often possible to rewrite ODEs of higher order as first-order ODEs by stacking the solution and the first $\nu - 1$ derivatives into a *state vector*. We write Equation (2.1) in state-space form as

$$\frac{\mathrm{d}}{\mathrm{d}t} \underbrace{\begin{pmatrix} x(t) \\ x^{(1)}(t) \\ \vdots \\ x^{(\nu-1)}(t) \end{pmatrix}}_{\text{state } \boldsymbol{x}(t)} = \begin{pmatrix} x^{(1)}(t) \\ \vdots \\ x^{(\nu-1)}(t) \\ f(t, x(t), x^{(1)}(t), \ldots, x^{(\nu-1)(t)}) \end{pmatrix}. \tag{2.2}$$

Example 1. *ODE model for a* pendulum *with a rod of length L.*

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\theta}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} \theta(t) \\ \dot{\theta}(t) \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ -\mu\dot{\theta} - (\frac{g}{L}\sin(\theta)) \end{pmatrix}. \tag{2.3}$$

μ is a "resistance" parameter (air resistance, friction, etc.) and $g \approx 9.81$ denotes the gravity of Earth.

We will in the remainder of this work often write first-order ODEs, using bold-faced notation for the state vector $\boldsymbol{x}(t)$, when talking about general ODEs.

Before elaborating on how to solve ODEs it is useful to first introduce the following setup. Let

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = f(t, \boldsymbol{x}(t)), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{2.4}$$

be an *initial value problem* (IVP) with a general vector field f. We consider the solution of the IVP on a bounded time interval $[t_0, t_{\max}]$. Specifying *initial conditions* $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ pins down the starting point of the solution. If no such conditions are provided, the ODE solution is only defined up to a set of additive constants. This is called the *general solution* (Särkkä and Solin, 2019). It is also possible to specify terminal conditions $\boldsymbol{x}(t_{\max})$, which gives a *boundary value problem* (BVP). Solving BVPs differs from solving IVPs in many regards, and is out of scope for this work.

Even with given initial conditions, for general f it is not clear whether a solution of Equation (2.4) exists and – if it exists – whether or not it is unique. Besides the considered time interval and the initial conditions, the existence of a (unique) solution

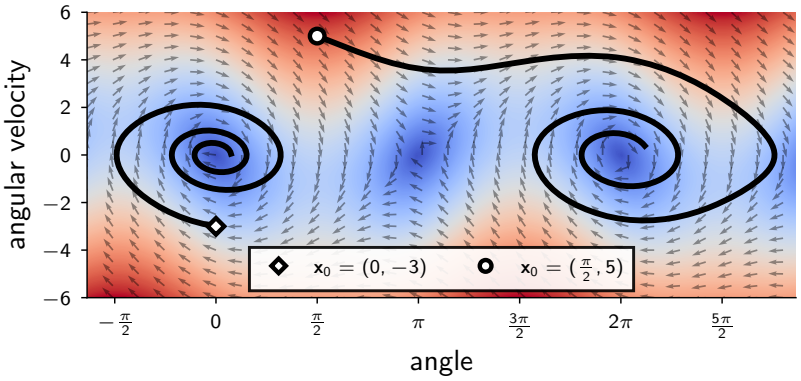


Figure 2.1: **Solving the same ODE with different initial conditions**. Depicted are two solutions of a pendulum ODE in state space (see Example 1). The directions (arrows) and magnitudes (color) of the derivatives defined by the vector field f are shown in the background. Depending on the initial value, the solutions can look completely differently.

depends largely on the vector field f. By integrating both sides of Equation (2.4) from t_0 to t, we obtain

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \int_{t_0}^{t} f(\tau, \boldsymbol{x}(\tau))\, \mathrm{d}\tau. \tag{2.5}$$

The appearance of the solution itself inside the vector field makes this problem extremely hard and already suggests that only specific (categories of) ODEs are analytically solvable at all.

By the *Picard-Lindelöf theorem*, a unique solution of Equation (2.4) exists if the limit

$$\lim_{n\to\infty} \boldsymbol{\varphi}_n(t) = \boldsymbol{x}(t), \tag{2.6}$$

exists, where $\boldsymbol{\varphi}_i(t), i = 1, 2, \ldots$ are given by the *Picard iteration*

$$\begin{aligned} \boldsymbol{\varphi}_0(t) &= \boldsymbol{x}_0 \\ \boldsymbol{\varphi}_i(t) &= \boldsymbol{x}_0 + \int_{t_0}^{t} f(\tau, \boldsymbol{\varphi}_{i-1}(\tau))\, \mathrm{d}\tau. \end{aligned} \tag{2.7}$$

One can show that the Picard iteration converges to the unique solution (Equation (2.6)) if the vector field $f(t, \boldsymbol{x}(t))$ is continuous in both arguments and Lipschitz-continuous in the second argument (Särkkä and Solin, 2019).

We will now touch on an important category of ODEs, which will play a crucial role throughout this work.

2.1.2 Linear ODEs

An ODE is linear if the vector field is a linear combination of the involved derivatives, i.e.

$$\begin{aligned} x^{(\nu)}(t) &= \left[\sum_{q=0}^{\nu-1} a_q(t) x^{(q)}(t)\right] + b(t)u(t), \\ x^{(q)}(t_0) &= x_0^{(q)}, \quad q = 0, \dots, \nu - 1, \end{aligned} \tag{2.8}$$

where $u(t)$ is some input into the system that is independent of the derivatives. We will proceed by writing these equations in state-space form, as well. For that, define the *companion matrix* $\boldsymbol{A}$ and the matrix $\boldsymbol{B}$

$$\boldsymbol{A}(t) = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & 1 & \ddots & \vdots \\ \vdots & \cdots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 1 \\ a_0(t) & a_1(t) & \cdots & \cdots & a_{\nu-1}(t) \end{pmatrix}, \qquad \boldsymbol{B}(t) = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ b(t) \end{pmatrix} \tag{2.9}$$

We will call the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ *system matrices*. Equation (2.8) in state-space form becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{2.10}$$

In order to write down a solution for this kind of ODE, we first define the *transition matrix* $\boldsymbol{\Phi}(t, s)$ of the ODE that has to fulfill the following properties:

$$\frac{\partial}{\partial t}\boldsymbol{\Phi}(t, s) = \boldsymbol{A}(t)\boldsymbol{\Phi}(t, s), \tag{2.11}$$

$$\frac{\partial}{\partial s}\boldsymbol{\Phi}(t, s) = -\boldsymbol{\Phi}(t, s)\boldsymbol{A}(s), \tag{2.12}$$

$$\boldsymbol{\Phi}(t, s) = \boldsymbol{\Phi}(t, \tau)\boldsymbol{\Phi}(\tau, s), \tag{2.13}$$

$$\boldsymbol{\Phi}(t, s) = \boldsymbol{\Phi}^{-1}(s, t), \tag{2.14}$$

$$\boldsymbol{\Phi}(t, t) = \boldsymbol{I}. \tag{2.15}$$

In what Särkkä and Solin (2019) calls the *integrating factor method*, Equation (2.10) is rearranged and multiplied with the transition matrix $\boldsymbol{\Phi}(t_0, t)$ to obtain

$$\begin{aligned} & \boldsymbol{\Phi}(t_0, t)\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) - \boldsymbol{\Phi}(t_0, t)\boldsymbol{A}(t)\boldsymbol{x}(t) = \boldsymbol{\Phi}(t_0, t)\boldsymbol{B}(t)\boldsymbol{u}(t), \\ \Leftrightarrow \quad & \boldsymbol{\Phi}(t_0, t)\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) + \frac{\partial}{\partial t}\boldsymbol{\Phi}(t_0, t)\boldsymbol{x}(t) = \boldsymbol{\Phi}(t_0, t)\boldsymbol{B}(t)\boldsymbol{u}(t), \\ \Leftrightarrow \quad & \frac{\partial}{\partial t}\left[\boldsymbol{\Phi}(t_0, t)\boldsymbol{x}(t)\right] = \boldsymbol{\Phi}(t_0, t)\boldsymbol{B}(t)\boldsymbol{u}(t). \end{aligned} \tag{2.16}$$

The first step uses Equation (2.12). The final step uses the product rule for derivatives "backward".

Integrating both sides from t_0 to t yields

$$\begin{aligned}
&\int_{t_0}^{t} \frac{\partial}{\partial \tau} \left[\boldsymbol{\Phi}(t_0, \tau)\boldsymbol{x}(\tau)\right] \mathrm{d}\tau = \int_{t_0}^{t} \boldsymbol{\Phi}(t_0, \tau)\boldsymbol{B}(\tau)\boldsymbol{u}(\tau)\,\mathrm{d}\tau, \\
\Leftrightarrow \quad &\boldsymbol{\Phi}(t_0, t)\boldsymbol{x}(t) - \underbrace{\boldsymbol{\Phi}(t_0, t_0)}_{=\boldsymbol{I}}\boldsymbol{x}(t_0) = \int_{t_0}^{t} \boldsymbol{\Phi}(t_0, \tau)\boldsymbol{B}(\tau)\boldsymbol{u}(\tau)\,\mathrm{d}\tau, \\
\Leftrightarrow \quad &\boldsymbol{\Phi}(t_0, t)\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t_0, \tau)\boldsymbol{B}(\tau)\boldsymbol{u}(\tau)\,\mathrm{d}\tau, \\
\Leftrightarrow \quad &\boldsymbol{x}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t, t_0)\boldsymbol{\Phi}(t_0, \tau)\boldsymbol{B}(\tau)\boldsymbol{u}(\tau)\,\mathrm{d}\tau, \\
\Leftrightarrow \quad &\boldsymbol{x}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t, \tau)\boldsymbol{B}(\tau)\boldsymbol{u}(\tau)\,\mathrm{d}\tau.
\end{aligned} \tag{2.17}$$

Remark 2. *A linear ODE with input $\boldsymbol{u}(t)$ implicitly defines a linear (to be precise: affine) operator acting on $\boldsymbol{u}(t)$.*

$$\begin{aligned}
\boldsymbol{x}(t) &= \boldsymbol{\Phi}(t, t_0)\boldsymbol{x}(t_0) + \underbrace{\int_{t_0}^{t} \boldsymbol{\Phi}(t, \tau)\boldsymbol{B}(\tau)\boldsymbol{u}(\tau)\,\mathrm{d}\tau}_{\text{linear mapping of } \boldsymbol{u}(t)}, \\
&=: \mathcal{S}_{\boldsymbol{x}_0}\left[\boldsymbol{u}\right](t)
\end{aligned} \tag{2.18}$$

In most cases, the transition matrix has no known, closed-form expression. If this were possible, linear ODEs could be solved analytically. It turns out that for constant system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ there exists a general expression for the transition matrix, which makes this class of differential equations quite interesting to study. In later chapters, linear time-invariant systems will play a crucial role.

Linear Time Invariant ODEs

An ODE is linear and time-invariant (LTI) if it is linear and the system matrices are constant over time, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t). \tag{2.19}$$

Discarding the input $\boldsymbol{B}\boldsymbol{u}(t)$ for now, we insert this equation into the Picard iteration from Equation (2.7). The explicit recursion becomes

$$\begin{aligned}
\boldsymbol{\varphi}_0(t) &= \boldsymbol{x}_0, \\
\boldsymbol{\varphi}_1(t) &= \boldsymbol{x}_0 + \int_{t_0}^{t} \boldsymbol{A}\boldsymbol{x}_0 \,\mathrm{d}\tau = \boldsymbol{x}_0 + \boldsymbol{A}(t-t_0)\boldsymbol{x}_0, \\
\boldsymbol{\varphi}_2(t) &= \boldsymbol{x}_0 + \int_{t_0}^{t} \boldsymbol{A}\left(\boldsymbol{x}_0 + \boldsymbol{A}(\tau - t_0)\boldsymbol{x}_0\right) \mathrm{d}\tau = \boldsymbol{x}_0 + \boldsymbol{A}(t-t_0)\boldsymbol{x}_0 + \frac{\left(\boldsymbol{A}(t-t_0)\right)^2}{2}\boldsymbol{x}_0, \\
&\vdots \\
\boldsymbol{\varphi}_N(t) &= \left(\sum_{k=0}^{N} \frac{\left(\boldsymbol{A}(t-t_0)\right)^k}{k!}\right)\boldsymbol{x}_0.
\end{aligned} \tag{2.20}$$

The infinite recursion $N \to \infty$ is given by the *matrix exponential*

$$\exp(\boldsymbol{A}) = \sum_{k=0}^{\infty} \frac{\boldsymbol{A}^k}{k!}. \tag{2.21}$$

In particular, for LTI ODEs, the transition matrix $\boldsymbol{\Phi}(t,s)$ is given by $\exp\left(\boldsymbol{A}(t-s)\right)$, and thus the solution of Equation (2.19) is

$$\boldsymbol{x}(t) = \exp\left(\boldsymbol{A}(t-t_0)\right)\boldsymbol{x}(t_0) + \int_{t_0}^{t} \exp\left(\boldsymbol{A}(t-\tau)\right)\boldsymbol{B}\boldsymbol{u}(\tau)\,\mathrm{d}\tau. \tag{2.22}$$

The matrix exponential can be computed efficiently in many mathematical software frameworks. This makes linear, time-invariant ODEs quite practicable. However, this comes with the obvious limitation that dealing with LTI models is often insufficient when studying physical systems in reality. When it comes to more complex systems, we have to consider a different palette of tools, that of iterative, numerical algorithms.

2.1.3 Numerical Solutions of ODEs

In practice, it is often required to consider complex vector fields that describe ODEs that do not have an analytical solution. Let

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = f(t, \boldsymbol{x}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{2.23}$$

be an IVP with a general (possibly nonlinear) vector field f and let $\Delta t > 0$ be a small time increment. Integrating both sides from t to $t + \Delta t$ yields

$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \int_t^{t+\Delta t} f(\tau, \boldsymbol{x}(\tau)) \, \mathrm{d}\tau. \tag{2.24}$$

Assuming that we can compute the integral, it is possible to compute the ODE solution at a discrete set of points $\mathbb{T} := (t_1, \ldots, t_n)$. Unfortunately, in most cases, it is not known how to compute the integral. Instead, numerical ODE solvers approximate the integral, where the kind of approximation varies between different methods.

The arguably most simple method is to approximate

$$\int_t^{t+\Delta t} f(\tau, \boldsymbol{x}(\tau)) \, \mathrm{d}\tau \approx \Delta t \cdot f(t, \boldsymbol{x}(t)), \tag{2.25}$$

which is called *explicit (or forward) Euler method* (e.g., Särkkä and Solin, 2019). The numerical ODE solution $\hat{\boldsymbol{x}}(t_i)$, where $t_i \in \mathbb{T}$, can then be computed as

$$\hat{\boldsymbol{x}}(t_i) = \hat{\boldsymbol{x}}(t_{i-1}) + \Delta t \cdot f(t, \hat{\boldsymbol{x}}(t_{i-1})), \quad i = 1, \ldots, n, \tag{2.26}$$

starting at $\hat{\boldsymbol{x}}(t_0) = \boldsymbol{x}_0$.

Remark 3. *The time grid $\mathbb{T}$ does not have to be equispaced. If it is, the step size $\Delta t = t_i - t_{i-1}, i = 1, \ldots, n$ is constant. There exist algorithms that choose the step size* adaptively *based on (local) error estimates of the numerical ODE solution (Hairer et al., 1993), in which the time grid is chosen while solving the ODE. Explicit Euler with adaptive steps would then be written as*

$$\hat{\boldsymbol{x}}(t_i) = \hat{\boldsymbol{x}}(t_{i-1}) + (\Delta t)_{i-1} \cdot f(t, \hat{\boldsymbol{x}}(t_{i-1})), \quad i = 1, \ldots, n, \tag{2.27}$$

which we omit for notational simplicity.

Due to its crude approximation, explicit Euler is not very practical for complex, real-world problems. The step size has to be chosen extremely small in order to obtain reasonable numerical ODE solutions, which leads to very expensive computations. In order to decrease computational cost, there exist more elaborate algorithms that use more complex approximations to the integral in Equation (2.24). This way, the computations per step get slightly more complex while the number of steps needed to obtain good numerical solutions decreases significantly. A popular example is the class of *Runge-Kutta methods* (e.g., Hairer et al., 1993), which approximate the integral in Equation (2.24) by a quadrature rule. The integration interval is partitioned into a number of intermediate steps $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_K$ at which the quadrature nodes and weights are chosen. For instance, the fourth-order Runge-Kutta method (often abbreviated RK4) computes the solution

steps as

$$\tilde{\boldsymbol{x}}_1 = \hat{\boldsymbol{x}}(t_{i-1}), \tag{2.28}$$

$$\tilde{\boldsymbol{x}}_2 = \hat{\boldsymbol{x}}(t_{i-1}) + \frac{\Delta t}{2} f(t_{i-1}, \tilde{\boldsymbol{x}}_1), \tag{2.29}$$

$$\tilde{\boldsymbol{x}}_3 = \hat{\boldsymbol{x}}(t_{i-1}) + \frac{\Delta t}{2} f(t_{i-1} + \frac{\Delta t}{2}, \tilde{\boldsymbol{x}}_2), \tag{2.30}$$

$$\tilde{\boldsymbol{x}}_4 = \hat{\boldsymbol{x}}(t_{i-1}) + \Delta t f(t_{i-1} + \Delta t, \tilde{\boldsymbol{x}}_3), \tag{2.31}$$

$$\begin{aligned} \hat{\boldsymbol{x}}(t_i) = \hat{\boldsymbol{x}}(t_{i-1}), \\ + \frac{\Delta t}{6} \left(f(t_{i-1}, \tilde{\boldsymbol{x}}_1) + 2f(t_{i-1} + \frac{\Delta t}{2}, \tilde{\boldsymbol{x}}_2) + 2f(t_{i-1} + \frac{\Delta t}{2}, \tilde{\boldsymbol{x}}_3) + f(t + \Delta t, \tilde{\boldsymbol{x}}_4) \right). \end{aligned} \tag{2.32}$$

The range of numerical methods for ODEs is large and different algorithms suggest themselves for different use cases. For the sake of brevity, the class of *implicit* methods has not been addressed at all, here. The reader is referred to, e.g. , Wanner and Hairer (1996) for more details. Large parts of this work will revolve around numerical algorithms for solving differential equations systems. Numerical computations take on the task of approximating an unknown quantity – in this case, an unknown function – which entails an unavoidable error. Section 2.4 will introduce a probabilistic numerical method for the solution of ODEs, which acknowledges this error and yields a posterior probability distribution over every possible ODE solution under the assumed model.

2.1.4 The SIRD Model

This section introduces a specific mechanistic model that is based on a system of ODEs and can be used to model the spread of an infectious disease within a population. The *SIRD model* (e.g., Hethcote, 2000) formulates the transitions between **s**usceptible, **i**nfectious, **r**ecovered, and **d**eceased people as

$$\begin{aligned} \frac{\mathrm{d}S(t)}{\mathrm{d}t} &= -\frac{\beta(t)S(t)I(t)}{P}, \\ \frac{\mathrm{d}I(t)}{\mathrm{d}t} &= \frac{\beta(t)S(t)I(t)}{P} - \gamma(t)I(t) - \eta(t)I(t), \\ \frac{\mathrm{d}R(t)}{\mathrm{d}t} &= \gamma(t)I(t), \\ \frac{\mathrm{d}D(t)}{\mathrm{d}t} &= \eta(t)I(t), \end{aligned} \tag{2.33}$$

governed by contact rate $\beta(t) : [t_0, t_{\max}] \to [0, 1]$, recovery rate $\gamma(t) : [t_0, t_{\max}] \to [0, 1]$, and mortality rate $\eta(t) : [t_0, t_{\max}] \to [0, 1]$ (Figure 2.2). S, I, R, and D evolve over time,

but the total population P (as the sum of the compartments) is assumed to remain constant. This model will serve as an interesting showcase for experiments in later sections. We thereby explicitly allow the parameters to be functions of time instead of being constant over time, as is also often encountered in the literature. Chapter 3 will evaluate a method that infers a latent force that represents time-varying ODE parameters on an SIRD model modeling the COVID-19 pandemic.

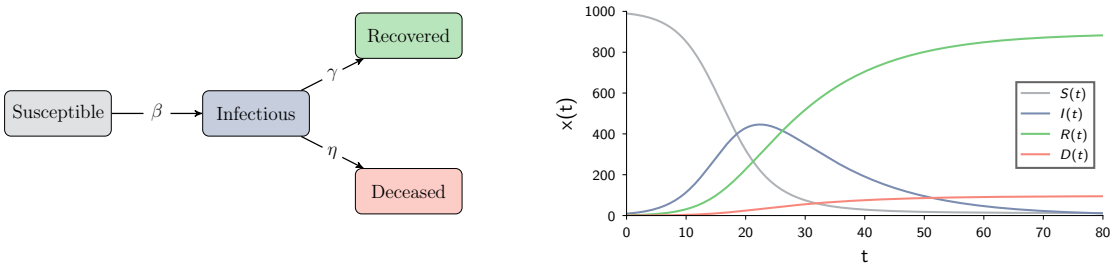


Figure 2.2: **The SIRD model.** On the left side, the transition of individuals between the compartments is depicted as a flow chart. The plot on the right shows the numerical solution (RK4) of an exemplary SIRD ODE with constant parameters $\beta(t) = 0.35, \gamma(t) = 0.07, \eta(t) = 0.0075$ and a population of $P = 1000$.

2.2 Probabilistic Inference in State-Space Models

This section considers a commonly occurring setup in which one aims to estimate an unknown state of a temporal process based on measurements thereof. We will approach this problem solely from a Bayesian viewpoint, deriving general equations for *Bayesian filtering*. The contents are mostly based on Särkkä (2013), to which the reader is referred for more detailed derivations.

The overarching goal is to infer a posterior probability distribution of the form

$$p(\boldsymbol{x}(t) \mid \mathcal{Y}(\tau)), \tag{2.34}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^d$ is the unknown state at time t, and $\mathcal{Y}(\tau)$ – roughly speaking – denotes a history of observations $(y(t_i))_{i=0}^N$, $y(t_i) \in \mathbb{R}^k$ related to the state $\boldsymbol{x}$ up until the time point τ. Depending on the time points t and τ, one can formulate mainly three different goals:

1. $t > \tau$: *Prediction.* Based on a history of observations, predict the future trajectory of the state $\boldsymbol{x}$.
2. $t = \tau$: *Filtering.* Estimate the current state of $\boldsymbol{x}$ based on past observations.
3. $t \leq \tau$: *Smoothing.* Having available a trajectory of observations over the entire considered time interval, estimate the state $\boldsymbol{x}$ on this interval.

As before, let $\mathbb{T} = (t_0, \ldots, t_N)$ denote a discrete set of time points. We will in the following simplify the notation in the sense that a sequence of states or observations at a consecutive sub-sequence of $\mathbb{T}$ will be denoted via a subscript of the form

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{x}(t_k) \\ \boldsymbol{y}_{k:l} &= (\boldsymbol{y}(t_k), \boldsymbol{y}(t_{k+1}), \ldots, \boldsymbol{y}(t_{l-1}), \boldsymbol{y}(t_l)), \quad \text{for } 1 \leq k < l \leq N. \end{aligned} \tag{2.35}$$

In the following we will set up a probabilistic state-space model, defining dynamics of the latent process and a measurement model under which said process is assumed to be observed.

2.2.1 Probabilistic State-Space Models

In the Bayesian sense, we begin by defining a prior belief over the unknown quantity. Concretely, let

$$p(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}) \tag{2.36}$$

denote the *transition density*, i.e. the stochastic dynamics of the latent process of interest. Since Equation (2.36) formalizes our belief of how the latent process $\boldsymbol{x}$ unfolds over time, it is also commonly called the *dynamics model*. It will turn out useful to consider only dynamics models of the form of Equation (2.36), in which the distribution over a state solely depends on the previous state. These processes are called *Markovian*, since they fulfill the *Markov property*

$$p(\boldsymbol{x}_n \mid \boldsymbol{x}_{0:n-1}) = p(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}). \tag{2.37}$$

In general, we will use the term "Markov" as a short form for "*first-order* Markov".

We proceed by defining a *measurement model* under which the latent process is assumed to be observed, as

$$p(\boldsymbol{y}_n \mid \boldsymbol{x}_n). \tag{2.38}$$

Together, Equations (2.36) and (2.38) define a *probabilistic state-space model*, for which

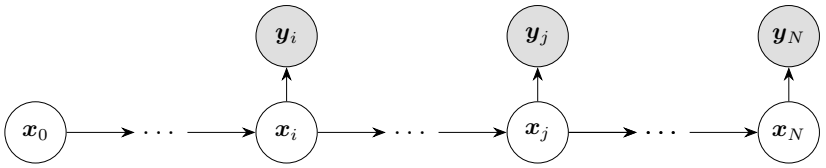


Figure 2.3: **Graphical model of the described probabilistic state-space model.** The Markov property can be read off from the chain-structure in $\boldsymbol{x}_{0:N}$.

a graphical model is provided in Figure 2.3. Besides the Markov property, from this

model one can also read off the following properties:

$$p(\boldsymbol{x}_n \mid \boldsymbol{x}_{0:n-1}, \boldsymbol{y}_{0:n-1}) = p(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}), \tag{2.39}$$

$$p(\boldsymbol{x}_n \mid \boldsymbol{x}_{n+1:N}, \boldsymbol{y}_{n+1:N}) = p(\boldsymbol{x}_n \mid \boldsymbol{x}_{n+1}), \tag{2.40}$$

$$p(\boldsymbol{y}_n \mid \boldsymbol{x}_{0:n}, \boldsymbol{y}_{0:n-1}) = p(\boldsymbol{y}_n \mid \boldsymbol{x}_n). \tag{2.41}$$

2.2.2 Bayesian Filtering and Smoothing

In order to compute the posterior from Equation (2.34), one could apply Bayes' rule using the state-space model equations. However, the posterior

$$p(\boldsymbol{x}_{0:N} \mid \boldsymbol{y}_{0:N}) = \frac{p(\boldsymbol{y}_{0:N}) \mid \boldsymbol{x}_{0:N}) p(\boldsymbol{x}_{0:N})}{p(\boldsymbol{y}_{0:N})} \tag{2.42}$$

is in general intractable to compute. This section will derive a recursive formula that makes use of the Markov property and the properties from Equations (2.39) to (2.41) in order to compute the filtering posterior in an online fashion, at linear cost in the number of data points.

General Bayesian Filtering Equations

The Bayesian filtering recursion can be split into two different steps: (i) the one-step-ahead *prediction* and (ii) the *measurement and correction* (often: *update*) step. Let the filtering distribution $p(\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n})$ at time point t_n be known. Then the prediction step is computed as

$$p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n}) = \int p(\boldsymbol{x}_{n+1}, \boldsymbol{x}_n \mid \boldsymbol{y}_{1:n}) \, \mathrm{d}\boldsymbol{x}_n \tag{2.43}$$

$$= \int p(\boldsymbol{x}_{n+1} \mid \boldsymbol{x}_n, \boldsymbol{y}_{1:n}) p(\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n}) \, \mathrm{d}\boldsymbol{x}_n \tag{2.44}$$

$$= \int \underbrace{p(\boldsymbol{x}_{n+1} \mid \boldsymbol{x}_n)}_{\text{dynamics model}} \quad \underbrace{p(\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n})}_{\text{filtering distribution}} \, \mathrm{d}\boldsymbol{x}_n. \tag{2.45}$$

The prediction step can thus be computed as an integral involving only the transition distribution (c.f. Equation (2.36)) and the filtering distribution at time t_n, which we assumed to be known. Equation (2.45) is known as the *Chapman-Kolmogorov Equation*.

Algorithm 1 Compute the Bayesian filtering posterior

Input: measurements $\boldsymbol{y}_{1:N}$, time grid $(t_0, \ldots, t_N)$.
Output: Filtering distribution $p(\boldsymbol{x}_N \mid \boldsymbol{y}_{1:N})$
Initialize $p(\boldsymbol{x}_0)$
for $n = 0, \ldots, N-1$ **do**
  Predict $p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n}) = \int p(\boldsymbol{x}_{n+1} \mid \boldsymbol{x}_n) p(\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n}) \, \mathrm{d}\boldsymbol{x}_n$
  Measure and correct $p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n+1}) \propto p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n}) p(\boldsymbol{y}_{n+1} \mid \boldsymbol{x}_{n+1})$
    (Normalize using Equation (2.49).
end for

The update step is then computed as

$$
\begin{aligned}
p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n+1}) &= p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n}, \boldsymbol{y}_{n+1}) && (2.46)\\
&\propto p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n}) p(\boldsymbol{y}_{n+1} \mid \boldsymbol{x}_{n+1}, \boldsymbol{y}_{1:n}) && (2.47)\\
&= \underbrace{p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n})}_{\text{prediction}} \quad \underbrace{p(\boldsymbol{y}_{n+1} \mid \boldsymbol{x}_{n+1})}_{\text{measurement model}} && (2.48)
\end{aligned}
$$

From Equation (2.46) to Equation (2.47), Bayes' rule was applied, neglecting a normalization factor that is required to obtain a valid posterior and can be computed as

$$
\int p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n}) p(\boldsymbol{y}_{n+1} \mid \boldsymbol{x}_{n+1}) \, \mathrm{d}\boldsymbol{x}_{n+1}. \tag{2.49}
$$

In order to compute the filtering distribution, the update step uses only the prediction step from Equation (2.45) and the measurement model (c.f. Equation (2.38)).

Algorithm 1 summarizes the Bayesian filtering algorithm in pseudocode.

Filtering in Linear Gaussian State-Space Models

Computing the normalized filtering posterior still involves complicated integrals in every step of the filtering algorithm. For general prior measures and likelihood functions, this is unavoidable. However, choosing linear Gaussian dynamics and measurements that are assumed to be corrupted by additive Gaussian i.i.d. noise, turns every step into closed-form Gaussian inference. This makes the algorithm computational very efficient.

Concretely, the following state-space model puts Equations (2.36) and (2.38) in concrete terms, by specifying linear Gaussian transitions and measurements as

$$
\begin{aligned}
\boldsymbol{x}_n &\sim \mathcal{N}\left(\boldsymbol{\Phi}_{n-1} \boldsymbol{x}_{n-1}, \boldsymbol{Q}_{n-1}\right), && (2.50)\\
\boldsymbol{y}_n &\sim \mathcal{N}\left(\boldsymbol{H}_n \boldsymbol{x}_n, \boldsymbol{R}_n\right), && (2.51)
\end{aligned}
$$

for transition matrix $\boldsymbol{\Phi}_{n-1} \in \mathbb{R}^{d \times d}$ and process noise covariance $\boldsymbol{Q}_{n-1} \in \mathbb{R}^{d \times d}$ at time t_{n-1}, as well as measurement matrix $\boldsymbol{H}_n \in \mathbb{R}^{k \times d}$ and measurement noise covariance

$\boldsymbol{R}_n \in \mathbb{R}^{k\times k}$ at time t_n. The following derivations do work for more general, affine transformations, which is neglected here to simplify notation. Due to Gaussians being closed under multiplication, this setup preserves the possibility to operate purely on Gaussians, which simplifies computation drastically. In particular, every step of the filtering procedure can be carried out in closed form, using only linear algebra operations.

Let $\boldsymbol{m}_n \in \mathbb{R}^d$ and $\boldsymbol{C}_n \in \mathbb{R}^{d\times d}$ denote mean vector and covariance matrix of the filtering distribution at time point t_n. We derive the moments of the Gaussian filtering distribution at the subsequent step by substituting the general distributions in the Bayesian filtering equations by Gaussians: For the prediction step we obtain

$$
\begin{aligned}
p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n}) &= \int p(\boldsymbol{x}_{n+1} \mid \boldsymbol{x}_n) p(\boldsymbol{x}_n \mid \boldsymbol{y}_{1:n}) \,\mathrm{d}\boldsymbol{x}_n && (2.52)\\
&= \int \mathcal{N}(\boldsymbol{x}_{n+1}; \boldsymbol{\Phi}_n \boldsymbol{x}_n, \boldsymbol{Q}_n) \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{m}_n, \boldsymbol{C}_n) \,\mathrm{d}\boldsymbol{x}_n && (2.53)\\
&= \int \mathcal{N}\left(\begin{pmatrix} \boldsymbol{x}_n \\ \boldsymbol{x}_{n+1} \end{pmatrix}; \begin{pmatrix} \boldsymbol{m}_n \\ \boldsymbol{\Phi}_n \boldsymbol{m}_n \end{pmatrix}, \begin{pmatrix} \boldsymbol{C}_n & \boldsymbol{C}_n \boldsymbol{\Phi}_n^\top \\ \boldsymbol{\Phi}_n \boldsymbol{C}_n & \boldsymbol{\Phi}_n \boldsymbol{C}_n \boldsymbol{\Phi}_n^\top + \boldsymbol{Q}_n \end{pmatrix} \right) \mathrm{d}\boldsymbol{x}_n && (2.54)\\
&= \mathcal{N}\left(\boldsymbol{x}_{n+1}; \boldsymbol{\Phi}_n \boldsymbol{m}_n, \boldsymbol{\Phi}_n \boldsymbol{C}_n \boldsymbol{\Phi}_n^\top + \boldsymbol{Q}_n\right) && (2.55)\\
&=: \mathcal{N}\left(\boldsymbol{x}_{n+1}; \boldsymbol{m}_{n+1}^-, \boldsymbol{C}_{n+1}^-\right), && (2.56)
\end{aligned}
$$

where the joint distribution of Gaussian random variables and closed-form Gaussian marginalization were used. The final step defines $\boldsymbol{m}_{n+1}^- \in \mathbb{R}^d$ and $\boldsymbol{C}_{n+1}^- \in \mathbb{R}^{d\times d}$ as the predicted mean and covariance at time point t_{n+1}. The prediction distribution is Gaussian, whereby mean and covariance are computed solely by linear and affine transformations of the previous filtering moments.

For the update step, closed-form Gaussian inference can be used to obtain

$$
\begin{aligned}
p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n}) &= \mathcal{N}\left(\boldsymbol{x}_{n+1}; \boldsymbol{m}_{n+1}^-, \boldsymbol{C}_{n+1}^-\right), && (2.57)\\
p(\boldsymbol{y}_{n+1} \mid \boldsymbol{x}_{n+1}, \boldsymbol{y}_{1:n}) &= p(\boldsymbol{y}_{n+1} \mid \boldsymbol{x}_{n+1}) = \mathcal{N}\left(\boldsymbol{y}_n; \boldsymbol{H}_{n+1} \boldsymbol{x}_{n+1}, \boldsymbol{R}_{n+1}\right), && (2.58)\\
p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n+1}) &= \mathcal{N}\left(\boldsymbol{x}_{n+1}; \boldsymbol{m}_{n+1}, \boldsymbol{C}_{n+1}\right), && (2.59)
\end{aligned}
$$

where

$$
\boldsymbol{m}_{n+1} = \boldsymbol{m}_{n+1}^- + \boldsymbol{C}_{n+1}^- \boldsymbol{H}_{n+1}^\top \left(\boldsymbol{H}_{n+1} \boldsymbol{C}_{n+1}^- \boldsymbol{H}_{n+1}^\top + \boldsymbol{R}_{n+1}\right)^{-1} \left(\boldsymbol{y}_{n+1} - \boldsymbol{H}_{n+1} \boldsymbol{m}_{n+1}^-\right), \tag{2.60}
$$

$$
\boldsymbol{C}_{n+1} = \boldsymbol{C}_{n+1}^- - \boldsymbol{C}_{n+1}^- \boldsymbol{H}_{n+1}^\top \left(\boldsymbol{H}_{n+1} \boldsymbol{C}_{n+1}^- \boldsymbol{H}_{n+1}^\top + \boldsymbol{R}_{n+1}\right)^{-1} \boldsymbol{H}_{n+1} \boldsymbol{C}_{n+1}^-. \tag{2.61}
$$

Equations (2.52) to (2.56) and Equations (2.57) to (2.61) give the *Kalman filter* equations, which combine the Bayesian filtering equations for linear Gaussian state-space models. It is worth stressing that in this setting, all distributions remain Gaussian and all operations are based on matrix-matrix and matrix-vector multiplication and addition. Both allow for simple and efficient implementation and hence render state-space

Algorithm 2 Kalman filter

Input: measurements $\boldsymbol{y}_{1:N}$, time grid $(t_0, \ldots, t_N)$, initial filtering moments $\boldsymbol{m}_0, \boldsymbol{C}_0$.
Output: Filtering distributions $\left(p(\boldsymbol{x}_i \mid \boldsymbol{y}_{1:i}) = \mathcal{N}\left(\boldsymbol{x}_i; \boldsymbol{m}_i, \boldsymbol{C}_i\right)\right)_{i=1}^{N}$
Initialize $\mathcal{N}(\boldsymbol{x}_0; \boldsymbol{m}_0, \boldsymbol{C}_0)$
for $n = 0, \ldots, N-1$ **do**
Predict

$$\boldsymbol{m}_{n+1}^{-} = \boldsymbol{\Phi}_n \boldsymbol{m}_n, \tag{2.62}$$
$$\boldsymbol{C}_{n+1}^{-} = \boldsymbol{\Phi}_n \boldsymbol{C}_n \boldsymbol{\Phi}_n^{\top} + \boldsymbol{Q}_n. \tag{2.63}$$

Measure and correct

$$\boldsymbol{r}_{n+1} = \boldsymbol{y}_{n+1} - \boldsymbol{H}_{n+1} \boldsymbol{m}_{n+1}^{-}, \tag{2.64}$$
$$\boldsymbol{S}_{n+1} = \boldsymbol{H}_{n+1} \boldsymbol{C}_{n+1}^{-} \boldsymbol{H}_{n+1}^{\top} + \boldsymbol{R}_{n+1}, \tag{2.65}$$
$$\boldsymbol{K}_{n+1} = \boldsymbol{C}_{n+1}^{-} \boldsymbol{H}_{n+1}^{\top} \boldsymbol{S}_{n+1}^{-1}, \tag{2.66}$$
$$\boldsymbol{m}_{n+1} = \boldsymbol{m}_{n+1}^{-} + \boldsymbol{K}_{n+1} \boldsymbol{r}_{n+1}, \tag{2.67}$$
$$\boldsymbol{C}_{n+1} = \boldsymbol{C}_{n+1}^{-} - \boldsymbol{K}_{n+1} \boldsymbol{S}_{n+1} \boldsymbol{K}_{n+1}^{\top}. \tag{2.68}$$

$p(\boldsymbol{x}_{n+1} \mid \boldsymbol{y}_{1:n+1}) = \mathcal{N}\left(\boldsymbol{m}_{n+1}, \boldsymbol{C}_{n+1}\right)$
end for

inference feasible even for many data points. Replacing the equations of Algorithm 1 accordingly yields the *Kalman filter* algorithm (Kalman, 1960) that is summarized in Algorithm 2.

Remark 4. *At every time point, the filtering distribution only contains data information from past and present measurements. After having computed the filtering posterior it is possible to obtain the* smoothing *posterior by carrying out another linear-time backward pass. This posterior then incorporates the information from the entire set of measurements at every time point. The smoothing-counterpart to the Kalman filter is called* Rauch-Tung-Striebel smoother *and is defined as a backward recursion starting at the final time step t_N, at which the smoothing moments $\boldsymbol{\xi}_N = \boldsymbol{m}_N$ and $\boldsymbol{\Lambda}_N = \boldsymbol{C}_N$ coincide with the filtering moments. The equations are given as (Särkkä, 2013)*

$$\boldsymbol{G}_{n-1} = \boldsymbol{C}_{n-1} \boldsymbol{\Phi}_{n-1}^{\top} \left[\boldsymbol{C}_n\right]^{-1}, \tag{2.69}$$
$$\boldsymbol{\xi}_{n-1} = \boldsymbol{m}_{n-1} + \boldsymbol{G}_{n-1} \left(\boldsymbol{\xi}_n - \boldsymbol{m}_n^{-}\right), \tag{2.70}$$
$$\boldsymbol{\Lambda}_{n-1} = \boldsymbol{C}_{n-1} + \boldsymbol{G}_{n-1} \left(\boldsymbol{\Lambda}_n - \boldsymbol{C}_n^{-}\right) \boldsymbol{G}_{n-1}^{\top}. \tag{2.71}$$

Filtering in Nonlinear Gaussian State-Space Models

In practice, it is often interesting to consider more general, nonlinear transitions. This turns the state-space model defined in Equations (2.50) and (2.51) into

$$\boldsymbol{x}_n \sim \mathcal{N}\left(\varphi(\boldsymbol{x}_{n-1}), \boldsymbol{Q}_{n-1}\right), \tag{2.72}$$
$$\boldsymbol{y}_n \sim \mathcal{N}\left(h(\boldsymbol{x}_n), \boldsymbol{R}_n\right), \tag{2.73}$$

for some transition function $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ and measurement function $h : \mathbb{R}^d \to \mathbb{R}^k$. While this setting is interesting in much more general applications, it does not immediately allow for efficient closed-form inference, as seen above.

The arguably most simple mitigation to this issue is to approximate the nonlinearities by a first-order Taylor approximation, linearizing at the predicted mean, which yields a linear Gaussian model again and standard Kalman filtering can be applied. This procedure is known as the *extended Kalman filter* (EKF) (Jazwinski, 1970; Maybeck, 1982) and is summarized by the following equations (c.f. Equations (2.62) to (2.68))

$$\boldsymbol{m}_{n+1}^{-} = \varphi(\boldsymbol{m}_n), \tag{2.74}$$
$$\boldsymbol{C}_{n+1}^{-} = \left[\mathrm{D}\varphi(\boldsymbol{m}_n)\right] \boldsymbol{C}_n \left[\mathrm{D}\varphi(\boldsymbol{m}_n)\right]^\top + \boldsymbol{Q}_n, \tag{2.75}$$
$$\boldsymbol{r}_{n+1} = \boldsymbol{y}_{n+1} - h(\boldsymbol{m}_{n+1}^{-}), \tag{2.76}$$
$$\boldsymbol{S}_{n+1} = \left[\mathrm{D}h(\boldsymbol{m}_{n+1}^{-})\right] \boldsymbol{C}_{n+1}^{-} \left[\mathrm{D}h(\boldsymbol{m}_{n+1}^{-})\right]^\top + \boldsymbol{R}_{n+1}, \tag{2.77}$$
$$\boldsymbol{K}_{n+1} = \boldsymbol{C}_{n+1}^{-} \left[\mathrm{D}h(\boldsymbol{m}_{n+1}^{-})\right]^\top \boldsymbol{S}_{n+1}^{-1}, \tag{2.78}$$
$$\boldsymbol{m}_{n+1} = \boldsymbol{m}_{n+1}^{-} + \boldsymbol{K}_{n+1}\boldsymbol{r}_{n+1}, \tag{2.79}$$
$$\boldsymbol{C}_{n+1} = \boldsymbol{C}_{n+1}^{-} - \boldsymbol{K}_{n+1}\boldsymbol{S}_{n+1}\boldsymbol{K}_{n+1}^\top, \tag{2.80}$$

where for a general function f we let $[\mathrm{D}f(x)]$ denote the Jacobian matrix of f evaluated at x. Plugging Equations (2.74) to (2.80) into Algorithm 2 yields the EKF algorithm. The EKF is categorized under *approximate Gaussian filtering* algorithms, which consist of further methods, as, for instance, the *unscented Kalman filter* (UKF) (Wan and Van Der Merwe, 2000; Julier and Uhlmann, 2004). Another class of methods that can be applied to the present setting are *sequential Monte Carlo methods* (Naesseth et al., 2019), which are not restricted to Gaussian distributions, but hence are significantly more costly to compute. This chapter refrains from going into detail about these and more available inference algorithms since they are not relevant for the remainder of this work. For details and a more exhaustive record of techniques for (approximate) state-space model inference, we refer to Särkkä (2013).

2.3 Linear Systems with Stochastic Input

This section takes up Section 2.1 in that it introduces linear systems in which we replace the input $\boldsymbol{u}(t)$ by a stochastic process. Besides being interesting for modeling unknown forces acting on a mechanistic system, this class of models will turn out to be useful for efficient Gaussian approximations of ODE solutions in later sections. The contents closely follow parts of Särkkä and Solin (2019).

2.3.1 Definitions

Definition 5. *Let $m : \mathbb{T} \to \mathbb{R}$ be any function, $k : \mathbb{T} \times \mathbb{T} \to \mathbb{R}$ be a Mercer kernel (Rasmussen and Williams, 2006). A* Gaussian process *$x \sim \mathcal{GP}(m, k)$ is a stochastic process such that every evaluation on a finite set of points $t_{1:n} = (t_1, \ldots, t_n) \subset \mathbb{T}$ is a (multivariate) Gaussian distributed random variable $x(t_{1:n}) \sim \mathcal{N}(m(t_{1:n}), k(t_{1:n}, t_{1:n}))$*

Definition 6. *A Gaussian* white noise *process $\dot{\mathbf{w}}(t)$ on $\mathbb{R}^d$ fulfills the following properties*

1. *For time points $t \neq t'$, $\dot{\mathbf{w}}(t)$ and $\dot{\mathbf{w}}(t')$ are independent.*

2. *The mapping $t \to \dot{\mathbf{w}}(t)$ is a Gaussian process with moments*

$$\mathbb{E}\left[\dot{\mathbf{w}}(t)\right] = \mathbf{0}, \tag{2.81}$$

$$\mathbb{C}\left[\dot{\mathbf{w}}(t), \dot{\mathbf{w}}(t')\right] = \mathbb{E}\left[\dot{\mathbf{w}}(t)\dot{\mathbf{w}}^\top(t')\right] = \delta(t - t')\boldsymbol{\Gamma}, \tag{2.82}$$

where δ denotes the Dirac delta and $\boldsymbol{\Gamma}$ is the spectral density of the white noise process (Särkkä and Solin, 2019).

We proceed by introducing *stochastic differential equations* (SDEs),

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = f(t, \mathbf{x}(t)) + \boldsymbol{B}(t, \mathbf{x}(t))\dot{\mathbf{w}}(t), \tag{2.83}$$

subject to Gaussian initial conditions

$$\mathbf{x}(t_0) \sim \mathcal{N}\left(\boldsymbol{m}_0, \boldsymbol{C}_0\right), \tag{2.84}$$

where the input $\dot{\mathbf{w}}(t)$ is a *white noise* process. Independence between $\mathbf{x}(t_0)$ and $\dot{\mathbf{w}}$ is always assumed.

Treatment of equations akin to Equation (2.83) differs fundamentally from ODEs with deterministic input. Most of the solution theory discussed in Section 2.1 is not applicable when it comes to SDEs, since the right-hand side is not continuous anymore due to the white noise input. In order to properly work with SDEs, we first rewrite Equation (2.83)

as an integral equation. Integrating both sides from t_0 to t gives

$$\mathbf{x}(t) - \mathbf{x}(t_0) = \int_{t_0}^{t} f(\tau, \mathbf{x}(\tau))\,\mathrm{d}\tau + \int_{t_0}^{t} \boldsymbol{B}(\tau, \mathbf{x}(\tau))\dot{\mathbf{w}}(\tau)\,\mathrm{d}\tau. \tag{2.85}$$

The right integral cannot be defined as a Riemann integral, since the white noise process under the integral is unbounded (Särkkä and Solin, 2019). Instead, it is commonly considered to be an *Itô integral*. Inside the Itô integral, we replace the increment $\dot{\mathbf{w}}(t)\,\mathrm{d}t$ by a *Brownian motion* $\mathrm{d}\mathbf{w}(t)$.

Definition 7. *A* Wiener process*, or* Brownian motion*,* $\mathbf{w}(t)$ *is a continuous stochastic process on* $\mathbb{R}^d$ *that fulfills the following properties:*

1. *For any* $0 \leq t_1 < t_2$*, the increment over the interval* $[t_1, t_2]$ *is a zero-mean Gaussian random variable*
$$\mathbf{w}(t_2) - \mathbf{w}(t_1) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma} \cdot |t_2 - t_1|),$$
where $\boldsymbol{\Gamma}$ *is the* diffusion matrix *of the Brownian motion.*
2. *If* $0 \leq t_1 < t_2 < \cdots < t_n < \infty$*, then the increments* $\mathbf{w}(t_m) - \mathbf{w}(t_{m-1}), 1 \leq m \leq n$ *are mutually independent.*
3. *The process starts at the origin:* $\mathbf{w}(0) = \mathbf{0}$*.*

Rewriting the equation in differential form then yields

$$\mathrm{d}\mathbf{x}(t) = f(t, \mathbf{x}(t))\,\mathrm{d}t + \boldsymbol{B}(t, \mathbf{x}(t))\,\mathrm{d}\mathbf{w}(t). \tag{2.86}$$

2.3.2 LTI SDEs and Gauss–Markov Processes

In this work, we solely consider linear time-invariant stochastic differential equations (LTI SDEs) (Øksendal, 2003; Särkkä and Solin, 2019). Similar to Equation (2.19), they take on the form

$$\mathrm{d}\mathbf{x}(t) = \boldsymbol{A}\mathbf{x}(t)\,\mathrm{d}t + \boldsymbol{B}\,\mathrm{d}\mathbf{w}(t), \quad \mathbf{x}(t_0) \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{C}_0), \tag{2.87}$$

where the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are constant. $\boldsymbol{A}$ and $\boldsymbol{B}$ are often called the *drift matrix* and the *dispersion matrix*, respectively.

From Remark 2 we know that the differential equation defines a linear operator acting on its input. Since GPs are closed under linear operators (Rasmussen and Williams, 2006) and the input is a Wiener process (and thus, a GP), we know that the solution of every linear SDE is a Gaussian process. Furthermore, it turns out that the solution of Itô SDEs are GPs with the Markov property (Särkkä and Solin, 2019). We call these processes *Gauss–Markov processes*. Examples include the Matérn, integrated Ornstein-Uhlenbeck, and integrated Wiener processes. Gauss–Markov processes are particularly

interesting since they imply a Gaussian transition density of the form

$$p(\mathbf{x}(t) \mid \mathbf{x}(s)) = \mathcal{N}\left(\mathbf{x}(t); \boldsymbol{\Phi}(t,s)\mathbf{x}(s), \boldsymbol{Q}(t,s)\right), \tag{2.88}$$

where the *process noise* covariance matrix $\boldsymbol{Q}(t,s)$ is given by

$$\boldsymbol{Q}(t,s) = \int_s^t \boldsymbol{\Phi}(t,\tau)\boldsymbol{B}\boldsymbol{B}^\top\boldsymbol{\Phi}^\top(t,\tau)\,\mathrm{d}\tau \tag{2.89}$$

This property of linear SDEs is extremely useful, since Equations (2.88) and (2.89) show that they can be rewritten as a *discrete* transition model, which allows for efficient state estimations. In particular, choosing Equation (2.87) as a prior over a latent process, we can write down a probabilistic state-space model that has exactly the form of Equations (2.50) and (2.51).

For LTI SDEs, the moments of the transition density are available in closed form and can be computed for instance with *matrix fraction decomposition* (Stengel, 1994; Axelsson and Gustafsson, 2015). Concretely, consider the solution to

$$\frac{\partial}{\partial t}\boldsymbol{\Psi}(t,s) = \begin{pmatrix} \boldsymbol{A}(t) & \boldsymbol{B}(t)\boldsymbol{B}(t)^\top \\ \mathbf{0} & -\boldsymbol{A}(t)^\top \end{pmatrix} \boldsymbol{\Psi}(t,s), \quad \boldsymbol{\Psi}(t,t) = \boldsymbol{I}. \tag{2.90}$$

Then

$$\boldsymbol{\Phi}(t,s) = \boldsymbol{\Psi}_{11}(t,s), \tag{2.91}$$

$$\boldsymbol{Q}(t,s) = \boldsymbol{\Psi}_{12}(t,s)\boldsymbol{\Phi}^\top(t,s). \tag{2.92}$$

As we here only consider the time-invariant case, the solution $\boldsymbol{\Psi}(t,s)$ can be computed with the matrix exponential as

$$\boldsymbol{\Psi}(t,s) = \boldsymbol{\Psi}(t-s) = \exp\left[\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}\boldsymbol{B}^\top \\ \mathbf{0} & -\boldsymbol{A}^\top \end{pmatrix}(t-s)\right], \tag{2.93}$$

and the moments of the transition density can be read off according to Equations (2.91) and (2.92). Should the transition density only depend on the difference between the time points, we will continue to write $\boldsymbol{\Phi}(t-s)$ and $\boldsymbol{Q}(t-s)$, respectively.

2.4 Probabilistic ODE Solvers

Section 2.1.3 explained how to numerically solve complex ODEs that do not have a known analytical solution. Thereby, we noticed that choosing a discrete time grid and the approximation of the integral from one time point to the next (Equation (2.24)) introduces a numerical error. Even though it is possible to reduce this error by choosing very small step sizes and elaborate approximations to the integral, it is never possible

to avoid the error on a finite-precision computer with finite computation time. Probabilistic numerics (PN) (Hennig et al., 2015; Cockayne et al., 2019; Wenger et al., 2021) concerns itself with quantifying these computational errors by phrasing computation as probabilistic inference. In this sense, probabilistic ODE solvers do not yield a trajectory that approximates the ODE solution with a point estimate, but instead return a posterior distribution over the solution, as shown in Figure 2.4. With Gaussian filtering (see Section 2.2) we availed ourselves of a language in which to phrase the computation of ODE solutions as probabilistic inference. This section sets up filtering-based probabilistic ODE solvers ("ODE filters") (Schober et al., 2019a; Tronarp et al., 2019; Kersting et al., 2020b; Tronarp et al., 2021; Bosch et al., 2021; Krämer and Hennig, 2020).

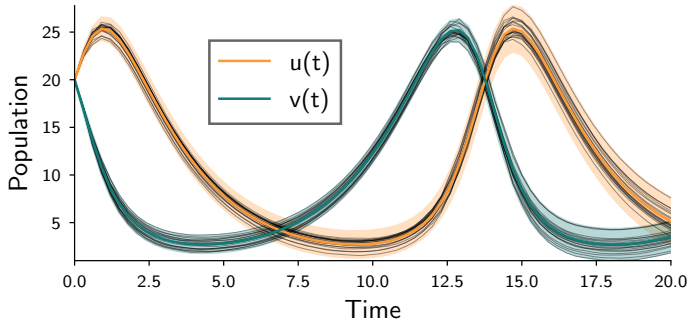


Figure 2.4: **Probabilistic ODE solution** of the Lotka-Volterra model (Lotka, 1978), describing the interaction between the populations of two species (predator $u(t)$ and prey $v(t)$) over time. The shaded areas depict the respective 90% confidence interval and the thin black lines are samples from the posterior solution. The example is adapted from the documentation of the `ProbNum` library (Wenger et al., 2021).

2.4.1 Prior Model

The following is common for ODE filters and parallels the presentation by e.g. Schober et al. (2019a).

Stochastic process prior on the ODE solution Let $\mathbf{x} := (\mathbf{x}^i)_{i=1}^d = (\mathbf{x}_0^i, \dots, \mathbf{x}_\nu^i)_{i=1}^d$ solve the LTI SDE

$$\mathrm{d}\mathbf{x}(t) = \boldsymbol{A}\mathbf{x}(t)\,\mathrm{d}t + \boldsymbol{B}\,\mathrm{d}\mathbf{w}(t), \tag{2.94}$$

subject to Gaussian initial conditions

$$\mathbf{x}(t_0) \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{C}_0) \tag{2.95}$$

for some $\boldsymbol{m}_0$ and $\boldsymbol{C}_0 := \boldsymbol{\Gamma} \otimes \breve{\boldsymbol{C}}_0$. The SDE is driven by a d-dimensional Wiener process $\mathbf{w}$ with diffusion $\boldsymbol{\Gamma} \in \mathbb{R}^{d \times d}$ and governed by the system matrices

$$\boldsymbol{A} := \boldsymbol{I}_d \otimes \breve{\boldsymbol{A}}, \quad \breve{\boldsymbol{A}} := \sum_{q=0}^{\nu-1} \boldsymbol{e}_q \boldsymbol{e}_{q+1}^\top, \quad \boldsymbol{B} := \boldsymbol{I}_d \otimes \boldsymbol{e}_\nu, \tag{2.96}$$

where $\boldsymbol{e}_q \in \mathbb{R}^{\nu+1}$ is the q-th basis vector. The zeroth component of $\mathbf{x}$, $(\mathbf{x}_0^i)_{i=1}^d$, is an integrated Wiener process. With such $\boldsymbol{A}$ and $\boldsymbol{B}$, the q-th component $(\mathbf{x}_q^i)_{i=1}^d$ models the q-th derivative of the integrated Wiener process. The canonical prior process for probabilistic ODE solvers are integrated Wiener processes (Schober et al., 2019b; Tronarp et al., 2019; Bosch et al., 2021; Krämer and Hennig, 2020). Similar SDEs can be written down for, e.g., the integrated Ornstein-Uhlenbeck process or the Matérn process (the only differences would be additional non-zero entries in $\boldsymbol{A}$). If $\boldsymbol{\Gamma}$ were diagonal, the Kronecker structure in $\boldsymbol{A}$ and $\boldsymbol{B}$ would imply prior pairwise independence between $\mathbf{x}^i$ and $\mathbf{x}^j$, $i \neq j$.

Discretization Let $\mathbb{T} = (t_0, ..., t_N)$ be some time-grid with step-size $h_n := t_{n+1} - t_n$. While for the presentation, we assume a fixed grid, practical implementations choose t_n adaptively. Reduced to $\mathbb{T}$, due to the Markov property, the process $\mathbf{x}$ becomes

$$\mathbf{x}(t_{n+1}) \mid \mathbf{x}(t_n) \sim \mathcal{N}(\boldsymbol{\Phi}(h_n)\mathbf{x}_n, \boldsymbol{Q}(h_n)) \tag{2.97}$$

for matrices $\boldsymbol{\Phi}(h_n)$ and $\boldsymbol{Q}(h_n)$, which are defined as

$$\boldsymbol{\Phi}(h_n) = \exp(\boldsymbol{A}h_n), \tag{2.98a}$$

$$\boldsymbol{Q}(h_n) = \int_0^{h_n} \boldsymbol{\Phi}(h_n - \tau)\boldsymbol{B}\boldsymbol{\Gamma}\boldsymbol{B}^\top \boldsymbol{\Phi}(h_n - \tau)^\top \, \mathrm{d}\tau. \tag{2.98b}$$

The definition of $\boldsymbol{\Phi}(h_n)$ uses the matrix exponential. $\boldsymbol{\Phi}(h_n)$ inherits the block diagonal structure from $\boldsymbol{A}$,

$$\boldsymbol{\Phi} := \boldsymbol{I}_d \otimes \breve{\boldsymbol{\Phi}}(h_n), \quad \breve{\boldsymbol{\Phi}}(h_n) = \exp(\breve{\boldsymbol{A}}h_n), \tag{2.99}$$

and $\boldsymbol{Q}$ has a Kronecker factorization similar to $\boldsymbol{C}_0$,

$$\boldsymbol{Q}(h_n) := \boldsymbol{\Gamma} \otimes \breve{\boldsymbol{Q}}(h_n), \tag{2.100a}$$

$$\breve{\boldsymbol{Q}}(h_n) := \int_0^{h_n} \breve{\boldsymbol{\Phi}}(h_n - \tau)\boldsymbol{e}_\nu \boldsymbol{e}_\nu^\top \breve{\boldsymbol{\Phi}}(h_n - \tau)^\top \, \mathrm{d}\tau. \tag{2.100b}$$

As detailed in Section 2.3.2, the discretization allows efficient extrapolation from t_n to t_{n+1}. Let $\mathbf{x}(t_n) \sim \mathcal{N}(\boldsymbol{m}_n, \boldsymbol{C}_n)$. Then,

$$\mathbf{x}(t_{n+1}) \sim \mathcal{N}(\boldsymbol{m}_{n+1}^-, \boldsymbol{C}_{n+1}^-) \tag{2.101}$$

with mean and covariance

$$\boldsymbol{m}_{n+1}^- = \boldsymbol{\Phi}(h_n)\boldsymbol{m}_n, \tag{2.102a}$$
$$\boldsymbol{C}_{n+1}^- = \boldsymbol{\Phi}(h_n)\boldsymbol{C}_n\boldsymbol{\Phi}(h_n)^\top + \boldsymbol{Q}(h_n). \tag{2.102b}$$

For improved numerical stability, probabilistic ODE solvers compute this prediction in square-root form, which means that only square-root matrices of $\boldsymbol{C}_n$ and $\boldsymbol{C}_{n+1}^-$ are propagated without ever forming full covariance matrices (Krämer and Hennig, 2020; Grewal and Andrews, 2014).

2.4.2 Information Model

Information operator The information operator

$$\mathcal{I}(\mathbf{x})(t) := \dot{\mathbf{x}}(t) - f(t, \mathbf{x}(t)), \tag{2.103}$$

known as the local defect (Gustafsson, 1992), captures "how well (a sample from) $\mathbf{x}$ solves the given ODE" – if this value is large, the current state is an inaccurate approximation, and if it is small, $\mathbf{x}$ provides a good estimate of the truth. The goal is to make the defect as small as possible over the entire time domain.

Artificial data The local defect $\mathcal{I}$ can be kept small by conditioning $\mathbf{x}$ on $\mathcal{I}(\mathbf{x})(t) \stackrel{!}{=} \mathbf{0}$ on "many" grid-points. Due to the regular prior and the regularity-preserving information operator $\mathcal{I}$, conditioning the prior on a zero-defect leads to an accurate ODE solution (Tronarp et al., 2021). Altogether, the probabilistic ODE solver targets the posterior distribution

$$p\left(\mathbf{x} \;\middle|\; \{\mathcal{I}(\mathbf{x})(t_n) = \mathbf{0}\}_{n=0}^N, \mathbf{x}_0(t_0) = \boldsymbol{y}_0\right). \tag{2.104}$$

(Recall from Equation (2.94) that lower indices in $\mathbf{x}$ refer to the derivative, i.e. $\mathbf{x}_0$ is the integrated Wiener process, and $\mathbf{x}_q$ its q-th derivative.) We call the posterior in Equation (2.104) the *probabilistic ODE solution*. Unfortunately, a nonlinear vector field f implies a nonlinear information operator $\mathcal{I}$. Thus, the exact posterior is intractable.

Linearization A tractable approximation of the probabilistic ODE solution is available through linearization. Linearizing f indirectly linearizes $\mathcal{I}$, and the corresponding probabilistic ODE solution arises via Gaussian inference. Let $\mathrm{D}_{\boldsymbol{x}} f(t, \boldsymbol{x})$ be (an approx-

imation of) the Jacobian of f with respect to $\boldsymbol{x}$. One can approximate the ODE vector field with a Taylor series

$$f(\boldsymbol{x}) \approx \hat{f}_{\boldsymbol{\xi}}(\boldsymbol{x}) := f(\boldsymbol{\xi}) + \mathrm{D}_{\boldsymbol{x}} f(\boldsymbol{\xi})(\boldsymbol{x} - \boldsymbol{\xi}) \tag{2.105}$$

at some $\boldsymbol{\xi} \in \mathbb{R}^d$. Let $\boldsymbol{E}_q := \boldsymbol{I}_d \otimes \boldsymbol{e}_q$ be the projection matrix that extracts the q-th derivative from the full state $\mathbf{x}$. In other words, $\dot{\mathbf{x}}_0 = \boldsymbol{E}_1 \mathbf{x}$. Equation (2.105) implies a linearization of $\mathcal{I}$ at some $\boldsymbol{\eta} \in \mathbb{R}^{d(\nu+1)}$,

$$\mathcal{I}(\mathbf{x})(t) \approx \hat{\mathcal{I}}_{\boldsymbol{\eta}}(\mathbf{x})(t) := \boldsymbol{H}(t)\mathbf{x}(t) + \boldsymbol{b}(t), \tag{2.106}$$

with linearization matrices

$$\boldsymbol{H}(t) := \boldsymbol{E}_1 - \mathrm{D}_{\boldsymbol{x}} f(t, \boldsymbol{E}_0 \boldsymbol{\eta}) \boldsymbol{E}_0, \tag{2.107a}$$

$$\boldsymbol{b}(t) := \mathrm{D}_{\boldsymbol{x}} f(t, \boldsymbol{E}_0 \boldsymbol{\eta}) \boldsymbol{E}_0 \boldsymbol{\eta} - f(t, \boldsymbol{E}_0 \boldsymbol{\eta}). \tag{2.107b}$$

$\hat{\mathcal{I}}_{\boldsymbol{\eta}}$ is linear in $\mathbf{x}(t)$. Therefore, the approximate probabilistic ODE solution becomes tractable with exact Gaussian filtering and smoothing once $\hat{\mathcal{I}}_{\boldsymbol{\eta}}$ is plugged into Equation (2.104) (Särkkä, 2013; Tronarp et al., 2019). As discussed in Section 2.2, at t_n, $\boldsymbol{\eta}$ is usually the predicted mean $\boldsymbol{m}_n^-$, which yields the extended Kalman filter. Choosing either (different) approximations of the Jacobian matrix $\mathrm{D}_{\boldsymbol{x}} f$, or using the exact, full Jacobian, yields different filtering-based probabilistic ODE solving algorithms. We consider three relevant versions:

1. *EK0:* Use the zero-matrix to approximate the Jacobian, $\mathrm{D}_{\boldsymbol{x}} f \equiv 0$, which has been a common choice since early work on ODE filters (Schober et al., 2019a; Kersting et al., 2020b), and implies a zeroth-order approximation of f (Tronarp et al., 2019).

2. *EK1:* Use the full Jacobian $\mathrm{D}_{\boldsymbol{x}} f = \nabla_{\boldsymbol{x}} f$, which amounts to a first-order Taylor approximation of the ODE vector field (Tronarp et al., 2019). In its general form, the EK1 does not fit the assumptions made below and thus does not immediately scale to high dimensions. Instead, we introduce the following variant:

3. *Diagonal EK1:* Use the diagonal of the full Jacobian, $\mathrm{D}_{\boldsymbol{x}} f = \mathrm{diag}(\nabla_{\boldsymbol{x}} f)$. This choice conserves the efficiency of the EK0 to a solver that uses Jacobian information. The diagonal EK1 is another minor contribution of the present work. The EK1 is more stable than the EK0 (Tronarp et al., 2019).

Measurement and correction A probabilistic ODE solver step consists of an extrapolation, measurement, and correction phase. Extrapolation has been explained in Equations (2.101) and (2.102) above. Denote

$$\mathbf{x}_{n+1}^- := \mathbf{x}(t_{n+1}) \sim \mathcal{N}(\boldsymbol{m}_{n+1}^-, \boldsymbol{C}_{n+1}^-). \tag{2.108}$$

The measurement phase approximates

$$\mathbf{z}_{n+1} := \hat{\mathcal{I}}_{\boldsymbol{m}^-_{n+1}}(\mathbf{x}^-_{n+1})(t_{n+1}) \approx \mathcal{I}(\mathbf{x}^-_{n+1})(t_{n+1}) \tag{2.109}$$

by exploiting $\boldsymbol{H}$ and $\boldsymbol{b}$,

$$\begin{aligned}
\mathbf{z}_{n+1} &\sim \mathcal{N}(\boldsymbol{z}_{n+1}, \boldsymbol{S}_{n+1}), && \text{(2.110a)}\\
\boldsymbol{z}_{n+1} &:= \boldsymbol{H}(t_{n+1})\boldsymbol{m}^-_{n+1} + \boldsymbol{b}(t_{n+1}), && \text{(2.110b)}\\
\boldsymbol{S}_{n+1} &:= \boldsymbol{H}(t_{n+1})\boldsymbol{C}^-_{n+1}\boldsymbol{H}(t_{n+1})^\top. && \text{(2.110c)}
\end{aligned}$$

$\mathbf{z}_{n+1}$ will be used for calibration (details below). The extrapolated random variable is then corrected as

$$\begin{aligned}
\mathbf{x}_{n+1} &\sim \mathcal{N}(\boldsymbol{m}_{n+1}, \boldsymbol{C}_{n+1}), && \text{(2.111a)}\\
\boldsymbol{m}_{n+1} &:= \boldsymbol{m}^-_{n+1} - \boldsymbol{C}^-_{n+1}\boldsymbol{H}(t_{n+1})^\top \boldsymbol{S}^{-1}_{n+1}\boldsymbol{z}_{n+1}, && \text{(2.111b)}\\
\boldsymbol{C}_{n+1} &:= \boldsymbol{\Xi}\,\boldsymbol{C}^-_{n+1}\,\boldsymbol{\Xi}^\top, && \text{(2.111c)}\\
\boldsymbol{\Xi} &:= \boldsymbol{I} - \boldsymbol{C}^-_{n+1}\boldsymbol{H}(t_{n+1})^\top \boldsymbol{S}^{-1}_{n+1}\boldsymbol{H}(t_{n+1}). && \text{(2.111d)}
\end{aligned}$$

The update in Equations (2.111c) and (2.111d) is the Joseph update (Bar-Shalom et al., 2004). In practice, we never form the full $\boldsymbol{C}_{n+1}$ but compute only the square-root matrix by applying $\boldsymbol{\Xi}$ to the square-root matrix of $\boldsymbol{C}^-_{n+1}$. It is not a Cholesky factor (because it is not lower triangular), but generic square-root matrices suffice for numerically stable implementation of probabilistic ODE solvers (Krämer and Hennig, 2020).

2.4.3 Square-Root Implementation of Probabilistic ODE Solvers

The following details the square-root implementation of the transitions underlying the probabilistic ODE solver. The whole section is taken from Supplement A in Krämer et al. (2021a) and is a synopsis of the explanations by Krämer and Hennig (2020). See also Grewal and Andrews (2014) for additional details.

Extrapolation

The extrapolation step

$$\boldsymbol{C}^-_{n+1} = \boldsymbol{\Phi}(h_n)\boldsymbol{C}_n\boldsymbol{\Phi}(h_n)^\top + \boldsymbol{Q}(h_n) \tag{2.112}$$

does not lead to stability issues further down the line (i.e. in calibration/correction/smoothing steps) if carried out in square-root form. Square-root form means that instead of tracking and propagating covariance matrices $\boldsymbol{C}$, only square-root matrices

$\boldsymbol{C} = \sqrt{\boldsymbol{C}}\sqrt{\boldsymbol{C}}^\top$ are used for extrapolation and correction steps without forming the full covariance. This is possible by means of QR decompositions.
Note: To avoid a clash of notation, we will denote the QR-factors with non-boldface letters Q and R. Recall that $\boldsymbol{Q}$ and $\boldsymbol{R}$ are reserved for the process noise covariance and the measurement noise covariance, respectively.

The matrix square root $\sqrt{\boldsymbol{C}_{n+1}^-}$ arises from $\sqrt{\boldsymbol{C}_n}$ through the QR decomposition of

$$Q\begin{pmatrix} R \\ 0 \end{pmatrix} \leftarrow \begin{pmatrix} \sqrt{\boldsymbol{C}_n}\boldsymbol{\Phi}(h_n)^\top \\ \sqrt{\boldsymbol{Q}(h_n)}^\top \end{pmatrix}, \quad \sqrt{\boldsymbol{C}_{n+1}^-} \leftarrow R^\top \tag{2.113}$$

because

$$\begin{aligned} R^\top R &= \begin{pmatrix} R \\ 0 \end{pmatrix}^\top Q^\top Q \begin{pmatrix} R \\ 0 \end{pmatrix} && (2.114)\\ &= \begin{pmatrix} \sqrt{\boldsymbol{C}_n}\boldsymbol{\Phi}(h_n)^\top \\ \sqrt{\boldsymbol{Q}(h_n)}^\top \end{pmatrix}^\top \begin{pmatrix} \sqrt{\boldsymbol{C}_n}\boldsymbol{\Phi}(h_n)^\top \\ \sqrt{\boldsymbol{Q}(h_n)}^\top \end{pmatrix} && (2.115)\\ &= \boldsymbol{\Phi}(h_n)\boldsymbol{C}_n\boldsymbol{\Phi}(h_n)^\top + \boldsymbol{Q}(h_n). && (2.116) \end{aligned}$$

QR decompositions of a rectangular matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$, $m > n$ costs $O(mn^2)$, which implies that the covariance square-root correction costs $O(d^3\nu^3)$. The QR decomposition is unique up to orthogonal row-operations (e.g. multiplying with ± 1). Probabilistic ODE solvers require only *any* square-root matrix, so this equivalence relation can be safely ignored – they all imply the same covariance.

Correction

The correction follows a similar pattern. Recall the linearized observation model

$$\mathcal{I}(\mathbf{x}) \approx \hat{\mathcal{I}}(\mathbf{x}) = \boldsymbol{H}\mathbf{x} + \boldsymbol{b} \tag{2.117}$$

where $\boldsymbol{H}$ contains the vector field information and (possibly) the Jacobian information. There are two ways of performing square-root corrections: the conventional way, and the way that is tailored to probabilistic ODE solvers, building on Joseph form corrections.

Conventional way Let $\sqrt{\boldsymbol{C}^-}$ be a matrix square root of the current extrapolated covariance (we drop the $n+1$ index for improved readability). Let $\mathbf{0}_n$ be the $n \times n$ zero matrix, and $\mathbf{0}_{n\times m}$ the $n \times m$ zero matrix. The heart of the square-root correction is another QR decomposition of the matrix

$$Q\begin{pmatrix} R_{11} & R_{12} \\ \mathbf{0}_{d(\nu+1)\times d} & R_{22} \end{pmatrix} \leftarrow \begin{pmatrix} \sqrt{\boldsymbol{C}^-}^\top \boldsymbol{H}^\top & \sqrt{\boldsymbol{C}^-}^\top \\ \mathbf{0}_{d(\nu+1)\times d} & \mathbf{0}_{d(\nu+1)} \end{pmatrix} \tag{2.118}$$

with $R_{11} \in \mathbb{R}^{d\times d}$, $R_{12} \in \mathbb{R}^{d\times d(\nu+1)}$, and $R_{22} \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$. The R_{ij} matrices contain the relevant information about the covariance matrices involved in the correction:

- $\sqrt{\boldsymbol{S}} = R_{11}^\top$ is the matrix square root of the innovation covariance
- $\sqrt{\boldsymbol{C}} = R_{22}^\top$ is the matrix square root of the posterior covariance
- $\boldsymbol{K} = R_{12}(R_{11})^{-1}$ is the Kalman gain and can be used to correct the mean

This QR decomposition costs $O(d^3\nu^3)$ again, but the matrix involved is larger than the stack of matrices in the extrapolation step (it has d more columns), so for high dimensional problems, the increased overhead becomes significant. However, if only *any* square-root matrix is desired, this step can be circumvented.

Joseph way Again, let $\sqrt{\boldsymbol{C}^-}$ be the square-root matrix of the current extrapolated covariance which results from the extrapolation step in square-root form. Next, the full covariance is assembled (which goes against the usual grain of avoiding full covariance matrices, but in the present case does the job) as $\boldsymbol{C}^- = \sqrt{\boldsymbol{C}^-}\sqrt{\boldsymbol{C}^-}^\top$. Since in the probabilistic solver, $\boldsymbol{C}^-$ is either a Kronecker matrix $\boldsymbol{I}_d \otimes \breve{\boldsymbol{C}}^-$ or a block diagonal matrix $\mathrm{blockdiag}((\boldsymbol{C}^-)^1, ..., (\boldsymbol{C}^-)^d)$, this is sufficiently cheap. The innovation covariance itself then becomes

$$\begin{aligned}
\boldsymbol{S} &= \boldsymbol{H}\boldsymbol{C}^-\boldsymbol{H}^\top && \text{(2.119a)}\\
&= (\boldsymbol{E}_1 - \mathrm{D}f(\mathbf{x})\boldsymbol{E}_0)\boldsymbol{C}^-(\boldsymbol{E}_1 - \mathrm{D}f(\mathbf{x})\boldsymbol{E}_0)^\top && \text{(2.119b)}\\
&= \boldsymbol{E}_1\boldsymbol{C}^-\boldsymbol{E}_1^\top - \mathrm{D}f(\mathbf{x})\boldsymbol{E}_0\boldsymbol{C}^-\boldsymbol{E}_1^\top - \boldsymbol{E}_1\boldsymbol{C}^-\boldsymbol{E}_0^\top\,[\mathrm{D}f(\mathbf{x})]^\top + \mathrm{D}f(\mathbf{x})\boldsymbol{E}_0\boldsymbol{C}^-\boldsymbol{E}_0^\top\,[\mathrm{D}f(\mathbf{x})]^\top && \text{(2.119c)}
\end{aligned}$$

which can be computed rather efficiently because $\boldsymbol{E}_i\boldsymbol{C}^-\boldsymbol{E}_j^\top$ only involves accessing elements, not matrix multiplication. The only non-negligible cost here is multiplication with the Jacobian of the ODE vector field (which is often sparse in high-dimensional problems). Then, the Kalman gain

$$\boldsymbol{K} = \boldsymbol{C}^-\boldsymbol{H}^\top\boldsymbol{S}^{-1} \tag{2.120}$$

can be computed from $\boldsymbol{S}$ which implies that the covariance correction reduces to

$$\sqrt{\boldsymbol{C}} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\sqrt{\boldsymbol{C}^-} \tag{2.121}$$

which is the "left half" of the Joseph correction. The resulting matrix is square, and a matrix square root of the posterior covariance, but not triangular thus no valid Cholesky factor. If the sole purpose of the square-root matrices is improved numerical stability, generic square-root matrices suffice.

2.4.4 Practical Considerations

Let us conclude with brief pointers to further practical considerations that are important for efficient probabilistic ODE solutions.

- *Initialization:* The ODE filter state models a stack of a state and the first ν derivatives. The stability of the probabilistic ODE solver depends on the accurate initialization of all derivatives. The current state of the art is to use Taylor-mode automatic differentiation (Krämer and Hennig, 2020; Griewank and Walther, 2008), whose complexity scales exponentially with the dimension of the ODE. Instead, we initialize the solver by inferring

$$p(\mathbf{x}(t_0) \mid \mathbf{x}_0(\tau_m) = \hat{\boldsymbol{x}}(\tau_m), m = 0, ..., \nu) \tag{2.122}$$

  on $\nu + 1$ small steps $\tau_0, ..., \tau_\nu$ where the $\hat{\boldsymbol{x}}(\tau_m)$ are computed with e.g. a Runge–Kutta method. This is a slight generalization of the strategy used by Schober et al. (2019a) (also refer to Schober et al. (2014); Gear (1980)), in the sense that we formulate this initialization as probabilistic inference instead of setting the first few means manually.

- *Error estimation:* Comprehensive explanation of error estimation and step-size adaptation is out of scope for the present work; we refer the reader to Schober et al. (2019a) and Bosch et al. (2021).

2.5 Partial Differential Equations

Partial differential equations (PDEs) are a widely-used way of modeling physical interdependencies between temporal and spatial variables. With the recent advent of physics-informed neural networks (Raissi et al., 2019), neural operators (Lu et al., 2021; Li et al., 2021), and neural ordinary/partial differential equations (Chen et al., 2018; Gelbrecht et al., 2021), PDEs have rapidly gained popularity in the machine learning community, too.

Let F, h, and g be given nonlinear functions and let $\Omega \subset \mathbb{R}^d$ be a sufficiently well-behaved domain with boundary $\partial\Omega$. $\mathcal{D}$ shall be a differential operator. The goal is to approximate an unknown function $\boldsymbol{u}$ that solves

$$\frac{\partial}{\partial t}\boldsymbol{u}(t, x) = F(t, x, \boldsymbol{u}(t, x), \mathcal{D}\boldsymbol{u}(t, x)), \tag{2.123}$$

for $t \in [t_0, t_{\max}]$ and $x \in \Omega$, subject to initial condition

$$\boldsymbol{u}(t_0, x) = h(x), \quad x \in \Omega, \tag{2.124}$$

and boundary conditions $\mathcal{B}\boldsymbol{u}(t,x) = g(x)$, $x \in \partial\Omega$. The differential operator $\mathcal{B}$ is usually the identity (Dirichlet conditions) or the derivative along normal coordinates (Neumann conditions). Except for only a few problems, PDEs do not admit closed-form solutions, and numerical approximations become necessary. This also affects machine learning strategies such as physics-informed neural networks or neural operators, for example, because they rely on fast generation of training data.

Definition 8. *A differential operator that occurs in many interesting PDEs is the* Laplace operator *(or Laplacian).*

$$\mathcal{D}\boldsymbol{x} = \Delta\boldsymbol{x} := \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2} \tag{2.125}$$

Example 9. *Section 2.1.4 introduced the SIRD system of ODEs. When studying the spread of infectious diseases across different regions, it is useful to specify prior mechanistic knowledge about the spatial diffusion of individuals between those regions. This can, for instance, be accomplished by adding a diffusion term, similar to Gai et al. (2020)*

$$\begin{aligned}
\frac{\partial S(t,x)}{\partial t} &= -\frac{\beta(t,x)S(t,x)I(t,x)}{P} + \rho_S \Delta S(t,x), \\
\frac{\partial I(t,x)}{\partial t} &= \frac{\beta(t,x)S(t,x)I(t,x)}{P} - \gamma(t,x)I(t,x) - \eta(t,x)I(t,x) + \rho_I \Delta I(t,x), \\
\frac{\partial R(t,x)}{\partial t} &= \gamma(t,x)I(t,x) + \rho_R \Delta R(t,x), \\
\frac{\partial D(t,x)}{\partial t} &= \eta(t,x)I(t,x),
\end{aligned} \tag{2.126}$$

for some diffusion coefficients ρ_S, ρ_I, ρ_R. These types of models are going to be considered briefly in Chapter 4.

2.5.1 Method of Lines

One common strategy for solving PDEs, called the *method of lines* (MOL; Schiesser, 2012), first discretizes the spatial domain Ω with a grid $\mathbb{X} := (x_0, ..., x_N)$, and then uses this grid to approximate the differential operator $\mathcal{D}$ with a matrix-vector product

$$(\mathcal{D}\boldsymbol{u})(t,\mathbb{X}) \approx \boldsymbol{D}\boldsymbol{u}(t,\mathbb{X}), \quad \boldsymbol{D} \in \mathbb{R}^{(N+1)\times(N+1)}, \tag{2.127}$$

where we use the notation $\boldsymbol{u}(t,\mathbb{X}) = (\boldsymbol{u}(t,x_n))_{n=0}^N$. Replacing the differential operator $\mathcal{D}$ with the matrix $\boldsymbol{D}$ turns the PDE into a system of ODEs. Standard ODE solvers can then numerically solve the resulting initial value problem.

A common choice for discretizing differential operators is the use of using finite-

difference approximations. The definition for the first-order derivative

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2.128}$$

suggests that the true derivative function $\frac{\mathrm{d}f(x)}{\mathrm{d}x}$ can be approximated by choosing a finite step size $h < \infty$ and evaluating the right-hand-side term. Constructing the Taylor expansion of $f(x+h)$ yields more finite-difference approximations that are accurate to higher orders. Further, using the definitions of higher-order derivatives, one can derive approximations for higher-order derivatives, as well (e.g. Table 1 in Fornberg (1988)).

As an example, consider the Laplacian from Definition 8 in one dimension. In order to approximate the operator in the *interior* of $\mathbb{X}$, $\operatorname{Int}\mathbb{X} = (x_1, \ldots, x_{N-1})$, we construct the matrix $\boldsymbol{D}_\Delta \in \mathbb{R}^{N-1 \times N-1}$ that contains the finite-difference weights as rows. Here, we assume that $\mathbb{X}$ consists of equispaced nodes with $|x_n - x_{n-1}| = h, \quad n = 1, \ldots, N$. For the Laplacian this results in the following banded matrix

$$\boldsymbol{D}_\Delta := \frac{1}{h^2} \cdot \begin{pmatrix} 1 & -2 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & -2 & 1 & 0 & \ddots & 0 \\ 0 & 0 & 1 & -2 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & \ldots & \ldots & 1 & -2 & 1 \end{pmatrix} \tag{2.129}$$

This yields the approximation from Equation (2.127) for $\mathcal{D} = \Delta$ on the interior of the discretization grid. Note that it is necessary to specify boundary conditions when discretizing a differential operator (which is neglected here by only building the matrix for the interior $\operatorname{Int}\mathbb{X}$). Chapter 4 will derive a probabilistic version of this method that tracks the numerical error that arises from the approximation in Equation (2.127) and provides details regarding the treatment of boundary conditions.

3 Joint Inference in Probabilistic State-Space Models

This chapter describes an algorithm that merges mechanistic knowledge in the form of an ODE with a non-parametric model over the parameters controlling the ODE – a *latent force* that represents quantities of interest. The algorithm then infers a trajectory that is informed by the observations but also follows sensible dynamics, as defined by the ODE, in the absence of observations. Algorithms for this purpose typically involve repeated forward simulations in the context of, e.g., Markov-chain Monte Carlo or optimization. The need for iterated computation of ODE solutions may demand simplifications in the model to meet limits in the computational budget. The main insight enabling our approach to this is that if probabilistic ODE solvers use the language of (extended) Kalman filters, conditioning on observations and solving the ODE itself is possible in one and the same process of Bayesian filtering and smoothing. Instead of iterated computation of ODE solutions, a posterior distribution arises from *a single* forward simulation, which has complexity equivalent to numerically computing an ODE solution, once, with a filtering-based, probabilistic ODE solver (Tronarp et al., 2019). Intuitively, one can think of this as opening up the black box ODE solver and acknowledging that each task – solving the ODE and discovering a latent force – is probabilistic inference in a state-space model.

3.1 Problem Setup

Let $\boldsymbol{x} : [t_0, t_{\max}] \to \mathbb{R}^d$ be a process that is observed at a discrete set of time points $\mathcal{T}_N^{\text{OBS}} := (t_0^{\text{OBS}}, ..., t_N^{\text{OBS}})$ through a sequence of measurements $\boldsymbol{y}_{0:N} := (\boldsymbol{y}_0, ..., \boldsymbol{y}_N) \in \mathbb{R}^{(N+1)\times k}$. Assume that these measurements are subject to additive i.i.d. Gaussian noise, according to the observation model

$$\mathbf{y}_n = \boldsymbol{H}\boldsymbol{x}(t_n) + \boldsymbol{\epsilon}_n, \quad \boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}), \tag{3.1}$$

for $n = 0, ..., N$ and matrices $\boldsymbol{H} \in \mathbb{R}^{k\times d}$ and $\boldsymbol{R} \in \mathbb{R}^{k\times k}$. Further suppose that $\boldsymbol{x}(t)$ solves the ODE

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t); \boldsymbol{u}(t)), \tag{3.2}$$

and satisfies the initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \in \mathbb{R}^d$. The vector field $f : \mathbb{R}^d \times \mathbb{R}^\ell \to \mathbb{R}^d$ is assumed to be autonomous, which is no loss of generality (e.g. (Krämer and Hennig,

2020)) but simplifies the notation. The *latent force* $\boldsymbol{u} : [t_0, t_{\max}] \to \mathbb{R}^\ell$ parametrizes f and shall be unknown.

Section 2.1.4 introduces the SIRD model that can be used to describe the spread of infectious diseases. This chapter will showcase the presented method on the COVID-19 pandemic, which is also commonly modeled using SIR-type models (e.g. (Giordano et al., 2020)). In this context, the contact rate $\beta(t)$ is the latent force and varies over time (in the notation from Equation (3.2), β is u). A time-varying contact rate provides a model for the impact of governmental measures on the dynamics of the pandemic. The experiments in Section 3.4 isolate the impact of the contact rate on the course of the infection counts, by assuming that γ and η are fixed and known. The method is by no means restricted to inference over a single latent force, as will also be shown in Section 3.4.1. In this SIRD setting, the goal is to infer an (approximate) joint posterior over $\beta(t)$ and the dynamics of $S(t)$, $I(t)$, $R(t)$, and $D(t)$ as well as to use the reconstructed dynamics to extrapolate into the future. Sections 3.2 and 3.3 explain the conceptual details, Section 3.4 evaluates the performance of the method and Section 3.5 distinguishes it from related work.

3.2 Model

3.2.1 Prior

Let $\nu \in \mathbb{N}$. Define two independent Gauss–Markov processes $\mathbf{u} : [t_0, t_{\max}] \to \mathbb{R}^\ell$ and $\mathbf{x} : [t_0, t_{\max}] \to \mathbb{R}^{d(\nu+1)}$ that solve the linear, time-invariant stochastic differential equations,

$$\begin{aligned} \mathrm{d}\mathbf{u}(t) &= \boldsymbol{A}_\mathbf{u}\mathbf{u}(t)\,\mathrm{d}t + \boldsymbol{B}_\mathbf{u}\,\mathrm{d}\mathbf{w}_\mathbf{u}(t), \\ \mathrm{d}\mathbf{x}(t) &= \boldsymbol{A}_\mathbf{x}\mathbf{x}(t)\,\mathrm{d}t + \boldsymbol{B}_\mathbf{x}\,\mathrm{d}\mathbf{w}_\mathbf{x}(t), \end{aligned} \tag{3.3}$$

with drift matrices $\boldsymbol{A}_\mathbf{u} \in \mathbb{R}^{\ell\times\ell}$ and $\boldsymbol{A}_\mathbf{x} \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$, as well as dispersion matrices $\boldsymbol{B}_\mathbf{u} \in \mathbb{R}^{\ell\times s}$ and $\boldsymbol{B}_\mathbf{x} \in \mathbb{R}^{d(\nu+1)\times d}$. $\mathbf{w}_\mathbf{u} : [t_0, t_{\max}] \to \mathbb{R}^s$ and $\mathbf{w}_\mathbf{x} : [t_0, t_{\max}] \to \mathbb{R}^d$ are Wiener processes. $\mathbf{u}$ and $\mathbf{x}$ satisfy the Gaussian initial conditions

$$\mathbf{u}(t_0) \sim \mathcal{N}(\boldsymbol{m}_\mathbf{u}, \boldsymbol{C}_\mathbf{u}), \quad \mathbf{x}(t_0) \sim \mathcal{N}(\boldsymbol{m}_\mathbf{x}, \boldsymbol{C}_\mathbf{x}), \tag{3.4}$$

defined by $\boldsymbol{m}_\mathbf{u} \in \mathbb{R}^\ell$, $\boldsymbol{C}_\mathbf{u} \in \mathbb{R}^{\ell\times\ell}$, $\boldsymbol{m}_\mathbf{x} \in \mathbb{R}^{d(\nu+1)}$, and $\boldsymbol{C}_\mathbf{u} \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$. $\mathbf{u}(t)$ models the unknown function $\boldsymbol{u}(t)$ and can be any Gauss–Markov process that admits a representation as the solution of a linear SDE with Gaussian initial conditions. $\mathbf{x}(t) = (\mathbf{x}^{(0)}(t), ..., \mathbf{x}^{(\nu)}(t)) \in \mathbb{R}^{d(\nu+1)}$ models the ODE dynamics, in light of which we require $\mathbf{x}^{(i)}(t) = \frac{\mathrm{d}^i}{\mathrm{d}t^i}\mathbf{x}^{(0)}(t) \in \mathbb{R}^d$, $i = 0, ..., \nu$. In other words, the first element in $\mathbf{x}(t)$ is an estimate for $\boldsymbol{x}(t)$, the second element is an estimate for $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t)$, et cetera. Encoding that the state $\mathbf{x}$ consists of a model for $\boldsymbol{x}(t)$ as well as its first ν derivatives imposes structure on $\boldsymbol{A}_\mathbf{x}$ and $\boldsymbol{B}_\mathbf{x}$ (see e.g. (Kersting et al., 2020b)).

Let $\Delta t > 0$. The transition densities of $\mathbf{u}$ and $\mathbf{x}$ are (Grewal and Andrews, 2014)

$$\begin{aligned} \mathbf{u}(t+\Delta t) \mid \mathbf{u}(t) &\sim \mathcal{N}(\boldsymbol{\Phi}_{\mathbf{u}}(\Delta t)\mathbf{u}(t), \boldsymbol{Q}_{\mathbf{u}}(\Delta t)), \\ \mathbf{x}(t+\Delta t) \mid \mathbf{x}(t) &\sim \mathcal{N}(\boldsymbol{\Phi}_{\mathbf{x}}(\Delta t)\mathbf{x}(t), \boldsymbol{Q}_{\mathbf{x}}(\Delta t)), \end{aligned} \tag{3.5}$$

where transition matrices $\boldsymbol{\Phi}_{\mathbf{u}}(\Delta t) \in \mathbb{R}^{\ell \times \ell}$ and $\boldsymbol{\Phi}_{\mathbf{x}}(\Delta t) \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$, as well as the process noise covariances $\boldsymbol{Q}_{\mathbf{u}}(\Delta t) \in \mathbb{R}^{\ell \times \ell}$ and $\boldsymbol{Q}_{\mathbf{x}}(\Delta t) \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$ are available in closed form and can be computed as described in Section 2.3.2.

The class of Gauss–Markov priors inherits its wide generalizability from Gaussian process models; recall that Gauss–Markov processes like $\mathbf{u}$ and $\mathbf{x}$ are Gaussian processes with the Markov property. While not every Gaussian process with one-dimensional input space is Markovian, a large number of descriptions of Gauss–Markov processes emerge by translating a covariance function into an (approximate) SDE representation (Särkkä and Solin, 2019, Chapter 12). For example, this applies to (quasi-)periodic, squared-exponential, or rational quadratic kernels; in particular, sums and products of Gauss–Markov processes admit a state-space representation (Solin and Särkkä, 2014; Särkkä and Solin, 2019). Recent research has considered approximate SDE representations of general Gaussian processes in one dimension (Loper et al., 2020). With these tools, prior knowledge over $\mathbf{u}$ or $\mathbf{x}$ can be encoded straightforwardly into the model.

3.2.2 Information Model

A functional relationship between the processes $\mathbf{u}(t)$, $\mathbf{x}(t)$ and the data $\boldsymbol{y}_{0:N}$ emerges by combining two likelihood functions: one for the observations $\boldsymbol{y}_{0:N}$ (recall Equation (3.1)), and one for the ordinary differential equation. The present section formalizes both. Let $\mathcal{T} = \mathcal{T}_N^{\text{OBS}} \cup \mathcal{T}_M^{\text{ODE}}$ be the union of the observation-grid $\mathcal{T}_N^{\text{OBS}}$, which has been introduced in Section 3.1, and an ODE-grid $\mathcal{T}_M^{\text{ODE}} := (t_0^{\text{ODE}}, ..., t_M^{\text{ODE}})$. The name "ODE-grid" expresses that this grid contains the locations on which the ODE information will enter the inference problem, as described below.

$\mathcal{T}_N^{\text{OBS}}$ contains the locations of $\boldsymbol{y}_{0:N}$, in light of which the first of two observation models is

$$\mathbf{y}_n \mid \mathbf{x}(t_n^{\text{OBS}}) \sim \mathcal{N}\left(\boldsymbol{H}\mathbf{x}^{(0)}(t_n^{\text{OBS}}), \boldsymbol{R}\right), \tag{3.6}$$

for $n = 0, \ldots, N$. This is a reformulation of the relationship between process $\mathbf{x}$ and observations $\mathbf{y}_{0:N}$ in Equation (3.1) in terms of $\mathbf{x}$ (instead of $\boldsymbol{x}$, which is modeled by $\mathbf{x}^{(0)}$). Including this first measurement model ensures that the inferred solution remains close to the data points. $\mathcal{T}_M^{\text{ODE}}$ contains the locations on which $\mathbf{u}(t)$ connects to $\mathbf{x}(t)$ through the ODE. Specifically, the set of random variables $\mathbf{z}_{0:M} \in \mathbb{R}^{(M+1)\times d}$, defined as

$$\mathbf{z}_m \mid \mathbf{x}(t_m^{\text{ODE}}), \mathbf{u}(t_m^{\text{ODE}}) \sim \delta\left(\mathbf{x}^{(1)}(t_m^{\text{ODE}}) - f\left(\mathbf{x}^{(0)}(t_m^{\text{ODE}}); \mathbf{u}(t_m^{\text{ODE}})\right)\right), \tag{3.7}$$

where δ is the Dirac delta, describes the discrepancy between the current estimate of the derivative of the ODE solution (i.e. $\mathbf{x}^{(1)}$) and its desired value (i.e. $f(\mathbf{x}^{(0)}; \mathbf{u})$), as prescribed by the vector field f. If the random variables $\mathbf{z}_{0:M}$ realize small values everywhere, $\mathbf{x}^{(0)}$ solves the ODE as parametrized by $\mathbf{u}$. This motivates introducing artificial data points $\boldsymbol{z}_{0:M} \in \mathbb{R}^{(M+1)\times d}$ that are equal to zero, $\boldsymbol{z}_m = \mathbf{0} \in \mathbb{R}^d$, $m = 0, ..., M$. Conditioning the stochastic processes $\mathbf{x}$ and $\mathbf{u}$ on attaining this (artificial) zero data ensures that the inferred solution follows ODE dynamics throughout the domain. Figure 3.1 shows the discretized state-space model.

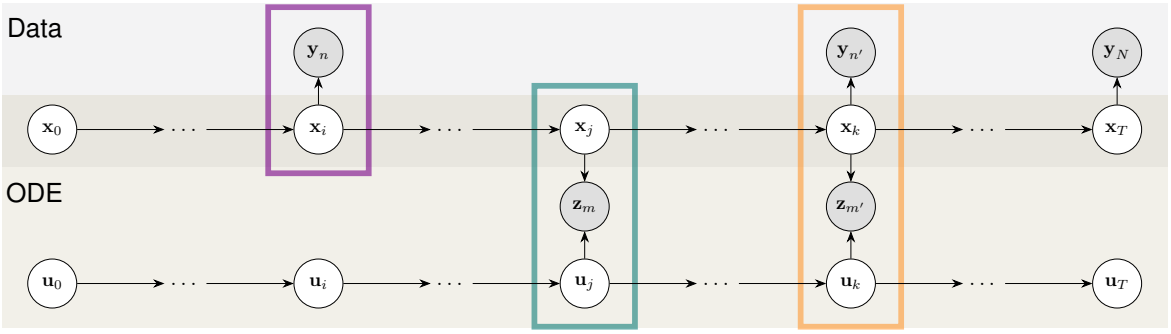


Figure 3.1: **Instance of the described state-space model, visualized as a directed graphical model**. Shaded variables are observed. Either *only data*, *only mechanistic knowledge*, or *both sources of information* can be conditioned on during inference.

3.3 Algorithm and Implementation

3.3.1 Augmented State-Space Model

First, for the processes $\mathbf{x}$ and $\mathbf{u}$, introduced in Section 3.2.1, an augmented state-space model is defined. The dynamics of the processes are modeled by the stochastic differential equation

$$\mathrm{d}\begin{pmatrix}\mathbf{u}(t)\\ \mathbf{x}(t)\end{pmatrix} = \underbrace{\begin{pmatrix}\boldsymbol{A}_\mathbf{u} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{A}_\mathbf{x}\end{pmatrix}}_{=:\boldsymbol{A}}\begin{pmatrix}\mathbf{u}(t)\\ \mathbf{x}(t)\end{pmatrix}\mathrm{d}t + \underbrace{\begin{pmatrix}\boldsymbol{B}_\mathbf{u} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{B}_\mathbf{x}\end{pmatrix}}_{=:\boldsymbol{B}}\mathrm{d}\begin{pmatrix}\mathbf{w}_\mathbf{u}(t)\\ \mathbf{w}_\mathbf{x}(t)\end{pmatrix}, \tag{3.8}$$

with Gaussian initial conditions

$$\begin{pmatrix}\mathbf{u}(t_0)\\ \mathbf{x}(t_0)\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}\boldsymbol{m}_\mathbf{u}(t_0)\\ \boldsymbol{m}_\mathbf{x}(t_0)\end{pmatrix}, \begin{pmatrix}\boldsymbol{C}_\mathbf{u}(t_0) & \mathbf{0}\\ \mathbf{0} & \boldsymbol{C}_\mathbf{x}(t_0)\end{pmatrix}\right). \tag{3.9}$$

The block-diagonal structure is due to the independent dynamics of the prior processes. The drift matrices $\boldsymbol{A}_\mathbf{u}$ and $\boldsymbol{A}_\mathbf{x}$, as well as the dispersion matrices $\boldsymbol{B}_\mathbf{u}$ and $\boldsymbol{B}_\mathbf{x}$ depend

on the choice of the respective processes $\mathbf{u}$ and $\mathbf{x}$. The measurement models are given in Equation (3.6) (for observed data) and in Equation (3.7) (for ODE measurements).

In the experiments presented in Sections 3.4.2 and 3.4.3 we model the latent contact rate $\beta(t)$ as a Matérn-$3/2$ process with characteristic length scale ℓ_q. Hence,

$$\mathrm{d}\mathbf{u}(t) = \underbrace{\begin{pmatrix} 0 & 1 \\ -\left(\sqrt{3}/\ell_q\right)^2 & -2\sqrt{3}/\ell_q \end{pmatrix}}_{\boldsymbol{A}_\mathbf{u}} \mathbf{u}(t)\,\mathrm{d}t + \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\boldsymbol{B}_\mathbf{u}} \mathrm{d}\mathbf{w}_\mathbf{u}(t). \tag{3.10}$$

The SIRD counts are modeled as the twice-integrated Wiener process

$$\mathrm{d}\mathbf{x}(t) = \underbrace{\begin{pmatrix} 0 & \boldsymbol{I}_d & 0 \\ 0 & 0 & \boldsymbol{I}_d \\ 0 & 0 & 0 \end{pmatrix}}_{\boldsymbol{A}_\mathbf{x}} \mathbf{x}(t)\,\mathrm{d}t + \underbrace{\begin{pmatrix} 0 \\ 0 \\ \boldsymbol{I}_d \end{pmatrix}}_{\boldsymbol{B}_\mathbf{x}} \mathrm{d}\mathbf{w}_\mathbf{x}(t), \tag{3.11}$$

such that $\mathbf{x} = \left(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right)^\top$ models the SIRD counts and the first two derivatives. Notice that $\boldsymbol{A}_\mathbf{x} \in \mathbb{R}^{d(\nu+1)\times d(\nu+1)}$ and $\boldsymbol{B}_\mathbf{x} \in \mathbb{R}^{d(\nu+1)\times d}$ are block matrices. $\boldsymbol{I}_d$ denotes the $d \times d$ identity matrix. In the context of the experiments, $d = 4$ (S, I, R, and D) and $\nu = 2$ (*twice*-integrated Wiener process). More details on the use of integrated Wiener processes in probabilistic ODE solvers can be found in, for instance, the work by Kersting et al. (2020b).

3.3.2 Approximate Inference

Both $\mathbf{x}$ and $\mathbf{u}$ enter the likelihood in Equation (3.7) through a possibly nonlinear vector field f. Therefore, the posterior distribution (recall $\boldsymbol{z}_{0:M} = \mathbf{0}$)

$$p\big(\mathbf{u}(t), \mathbf{x}(t) \mid \mathbf{z}_{0:M} = \boldsymbol{z}_{0:M},\ \mathbf{y}_{0:N} = \boldsymbol{y}_{0:N}\big) \tag{3.12}$$

is intractable, but can be approximated efficiently. Even though the problem is discretized, the posterior distribution is continuous (Särkkä and Solin, 2019, Chapter 10). There are mainly two approaches to computing a tractable approximation of the intractable posterior distribution in Equation (3.12): approximate Gaussian filtering and smoothing (Särkkä, 2013), which computes a cheap, Gaussian approximation of this posterior, and sequential Monte Carlo methods (Naesseth et al., 2019), whose approximate posterior may be more descriptive, but also more expensive to compute. Like the literature on probabilistic ODE solvers (Tronarp et al., 2019; Bosch et al., 2021), this work uses approximate Gaussian filtering and smoothing techniques for their low computational complexity.

The continuous-discrete state-space model inherits its nonlinearity from the ODE vector field f. Linearizing f with a first-order Taylor series expansion creates a tractable

Algorithm 3 Compute the filtering distribution by conditioning on both $y_{0:N}$ and $z_{0:M}$.

Input: data $\boldsymbol{y}_{0:N}$, time grid $\mathcal{T} = \mathcal{T}_N^{\text{OBS}} \cup \mathcal{T}_M^{\text{ODE}}$, vector field f, $\boldsymbol{m}_{\mathbf{x}}$, $\boldsymbol{C}_{\mathbf{x}}$, $\boldsymbol{m}_{\mathbf{u}}$, $\boldsymbol{C}_{\mathbf{u}}$
Output: Filtering distribution [Equation (3.13)]
Initialize $\mathbf{x}_0 = \mathcal{N}(\boldsymbol{m}_{\mathbf{x}}, \boldsymbol{C}_{\mathbf{x}})$ and $\mathbf{u}_0 = \mathcal{N}(\boldsymbol{m}_{\mathbf{u}}, \boldsymbol{C}_{\mathbf{u}})$ [Equation (3.4)]
for $t_j \in \mathcal{T}$ **do**
  Predict $\mathbf{x}_j$ from $\mathbf{x}_{j-1}$ and predict $\mathbf{u}_j$ from $\mathbf{u}_{j-1}$
  if $t_j \in \mathcal{T}_N^{\text{OBS}}$ **then**
    update $\mathbf{x}_j$ on $\boldsymbol{y}_j$ [Equation (3.6)]
  end if
  if $t_j \in \mathcal{T}_M^{\text{ODE}}$ **then**
    linearize measurement model and update $\mathbf{x}_j$ and $\mathbf{u}_j$ on $\boldsymbol{z}_j$ [Equation (3.7)]
  end if
end for

inference problem; more specifically, it gives rise to the extended Kalman filter (EKF; see Sections 2.2 and 2.4). Loosely speaking, if the random variable $\mathbf{z}$ is large in magnitude, then $\mathbf{x}$ and $\mathbf{u}$ are poor estimates for the ODE and its parameter. An EKF update, based on the first-order linearization of f, approximately corrects this misalignment. If sufficiently many ODE measurements $\boldsymbol{z}_{0:M}$ are available, a sequence of such updates preserves sensible ODE dynamics over time. An alternative to a Taylor-series linearization is the unscented transform, which yields the unscented Kalman filter (Wan and Van Der Merwe, 2000; Julier and Uhlmann, 2004). The computational complexity of both algorithms is linear in the number of grid points and cubic in the dimension of the state-space. Detailed implementation schemes can be found, for instance, in Särkkä (2013).

The EKF approximates the filtering distribution

$$p\left(\mathbf{u}(t), \mathbf{x}(t) \mid \mathbf{z}_{0:m} = \boldsymbol{z}_{0:m}, \mathbf{y}_{0:n} = \boldsymbol{y}_{0:n}, \text{ such that } t_m^{\text{ODE}}, t_n^{\text{OBS}} \leq t\right). \tag{3.13}$$

It describes the current state of the system given all the previous measurements and allows updates in an online fashion as soon as new observations emerge. If desired, the Rauch-Tung-Striebel smoother turns the filtering distribution into an approximation of the full (smoothing) posterior (in Equation (3.12)). In doing so, all observations – that is, measurements according to both Equation (3.6) and Equation (3.7) – are taken into account for the posterior distribution at each location t. As special cases, this setup recovers: (i) a Kalman filter/Rauch-Tung-Striebel smoother (Kalman, 1960) if the ODE likelihood (Equation (3.7)) is omitted; (ii) an ODE solver (Tronarp et al., 2019), if the data likelihood (Equation (3.6)) is omitted. In the present setting, however, both likelihoods play an important role.

The procedure is summarized in Algorithm 3. The prediction step is determined by the prior and is available in closed-form. As before, the predicted mean and covariance

are given as

$$\boldsymbol{m}_j^- = \boldsymbol{\Phi}(\Delta t)\, \boldsymbol{m}_{j-1}, \tag{3.14}$$
$$\boldsymbol{C}_j^- = \boldsymbol{\Phi}(\Delta t)\boldsymbol{C}_{j-1}\boldsymbol{\Phi}(\Delta t)^\top + \boldsymbol{Q}(\Delta t), \tag{3.15}$$

for given initial conditions $\boldsymbol{m}_0$, $\boldsymbol{C}_0$. The prediction step is the same, for both $t_j \in \mathcal{T}^{\text{OBS}}$ and $t_j \in \mathcal{T}^{\text{ODE}}$.

Two different update steps are defined for two kinds of observations. When observing data $\boldsymbol{y}_{0:N}$, i.e. $t_n \in \mathcal{T}^{\text{OBS}}$, the update step follows the rules of a standard Kalman filter. Concretely, the updated mean $\boldsymbol{m}_n$ and covariance $\boldsymbol{C}_n$ at time t_n are computed as in Equations (2.62) to (2.68)

$$\boldsymbol{r}_n = \boldsymbol{y}_n - \boldsymbol{H}\boldsymbol{m}_n^-, \tag{3.16}$$
$$\boldsymbol{S}_n = \boldsymbol{H}\boldsymbol{C}_n^-\boldsymbol{H}^\top + \boldsymbol{R}, \tag{3.17}$$
$$\boldsymbol{K}_n = \boldsymbol{C}_n^-\boldsymbol{H}^\top\boldsymbol{S}_n^{-1}, \tag{3.18}$$
$$\boldsymbol{m}_n = \boldsymbol{m}_n^- + \boldsymbol{K}_n\boldsymbol{v}_n, \tag{3.19}$$
$$\boldsymbol{C}_n = \boldsymbol{C}_n^- - \boldsymbol{K}_n\boldsymbol{S}_n\boldsymbol{K}_n^\top. \tag{3.20}$$

The matrices $\boldsymbol{H}$ and $\boldsymbol{R}$ are defined as in Equation (3.6).

Recall the ODE measurement model from Equation (3.7), which we here denote as h, as

$$h\left(\begin{pmatrix}\mathbf{u}(t)\\ \mathbf{x}(t)\end{pmatrix}\right) = \mathbf{x}^{(1)} - f\left(\mathbf{x}^{(0)}; \mathbf{u}(t)\right). \tag{3.21}$$

At locations $t_m \in \mathcal{T}^{\text{ODE}}$, we condition on the ODE measurements $\boldsymbol{z}_{0:M}$. Recall that these pseudo-observations are all zero. As in Equations (2.74) to (2.80),

$$\boldsymbol{r}_m = \boldsymbol{z}_m - h(\boldsymbol{m}_m^-), \tag{3.22}$$
$$\boldsymbol{S}_m = \left[\mathrm{D}h(\boldsymbol{m}_m^-)\right]\boldsymbol{C}_m^-\left[\mathrm{D}h(\boldsymbol{m}_m^-)\right]^\top + \lambda^2 I_d, \tag{3.23}$$
$$\boldsymbol{K}_m = \boldsymbol{C}_m^-\left[\mathrm{D}h(\boldsymbol{m}_m^-)\right]^\top\boldsymbol{S}_m^{-1}, \tag{3.24}$$
$$\boldsymbol{m}_m = \boldsymbol{m}_m^- + \boldsymbol{K}_m\boldsymbol{v}_m, \tag{3.25}$$
$$\boldsymbol{C}_m = \boldsymbol{C}_m^- - \boldsymbol{K}_m\boldsymbol{S}_m\boldsymbol{K}_m^\top, \tag{3.26}$$

where $[\mathrm{D}h(\boldsymbol{m}_m^-)]$ denotes the Jacobian of h at $\boldsymbol{m}_m^-$. In the case of a Dirac likelihood (see Equation (3.7)), $\lambda^2 = 0$ holds. For numerical stability (especially for $\lambda^2 = 0$) one can instead implement square-root filtering (see, e.g., (Grewal and Andrews, 2014; Krämer and Hennig, 2020)). All experiments in Section 3.4 use square-root filtering. The filtering distribution can be turned into a smoothing posterior by running a backwards-pass with a Rauch-Tung-Striebel smoother (e.g. (Särkkä, 2013)). The computational cost of obtaining either, the filtering or the smoothing posterior, are both linear in the number of grid points and cubic in the dimension of the state-space, i.e. $\mathcal{O}((N+M)(d^3\nu^3 + \ell^3))$.

Only a single forward-backward pass is required. If desired, the approximate Gaussian posterior can be refined iteratively by means of posterior linearization and iterated Gaussian filtering and smoothing, which yields the maximum-a-posteriori (MAP) estimate (Bell, 1994; Tronarp et al., 2018). The experiments presented in Section 3.4 show how a single forward-backward pass already approximates the MAP estimate accurately.

3.4 Experiments

This section describes three blocks of experiments. All experiments use a conventional, consumer-level CPU. First, a range of artificial datasets is generated by sampling ODE parameters from a prior state-space model and simulating a solution of the corresponding ODE. Inference in such a controlled environment allows comparing to the ground truth, thereby assessing the quality of the approximate inference. We consider three ODE models to this end. Second, a COVID-19 dataset will probe the predictive performance of the probabilistic model and the resulting approximate posterior distribution. Third, some changes to the model from the COVID-19 experiments, for instance, ensuring that the number of case counts must be positive, will improve the interpretability (for example, of the credible intervals). Controlling the range of values that the prior state-space can realize introduces additional nonlinearity into the model – which can also be locally approximated by the EKF – and makes the solution more physically meaningful.

3.4.1 Simulated Environments

As a first test for the capabilities of the proposed method, we consider three simulated environments. To this end, the training data is generated as follows. The starting point is always an initial value problem with dynamics defined by a vector field f and a Gauss–Markov prior over the dynamics $\mathbf{x}$ and the unknown parameters $\mathbf{u}$ of the vector field. Then, (i) we sample the time-varying parameter trajectories from the Gauss–Markov prior; (ii) we solve the ODE, as parametrized by the sampled trajectories from (i), using LSODA (Hindmarsh and Petzold, 2005) with adaptive step sizes using SciPy (Virtanen et al., 2020); (iii) we subsample the ground-truth solution on a uniform grid (which will become $\mathcal{T}_N^{\text{OBS}}$) to generate artificial state observations $\boldsymbol{y}_{0:N}$; (iv) we add Gaussian i.i.d. noise to the observations.

The procedure described above generates both a ground truth to compare to and a noisy, artificially observed data set. Given such a set of observations, Algorithm 3 computes a posterior distribution over the true trajectories under appropriate model assumptions. In this posterior, we look for the proximity of the mean estimate to the underlying ground truth; the closer, the better. We measure this proximity in the root-mean-square error. Furthermore, the width of the posterior (expressed by the posterior covariance) should deliver an appropriate quantification of the mismatch. We report the χ^2-statistic (Bar-Shalom et al., 2004), which suggests that the posterior distribution is

well-calibrated if the χ^2-statistic is close to the dimension d of the ground truth. Three mechanistic models serve as examples.

Van-der-Pol The first of three test problems is the van-der-Pol oscillator (Guckenheimer, 1980). It has one parameter μ (sometimes referred to as a stiffness constant, because for large μ, the van-der-Pol system is stiff). As a prior over the dynamics we choose a twice-integrated Wiener process with diffusion intensity $\sigma^2_{\mathbf{x}} = 300$. The stiffness parameter μ is modeled as a Matérn-$3/2$ process with lengthscale $\ell_{\mathbf{u}} = 10$ and diffusion intensity $\sigma^2_{\mathbf{u}} = 0.3$. The posterior is computed on a grid from $t_0 = 0$ to $t_{\max} = 25$ units of time with step size $\Delta t = 0.025$.

Lotka-Volterra The Lotka-Volterra equations (Lotka, 1978) describe the change in the size of two populations, predators and prey. There are four parameters, which we call a, b, c, and d, which describe the interaction and death/reproduction rates of the populations. As a prior over the dynamics we choose a twice-integrated Wiener process with diffusion intensity $\sigma^2_{\mathbf{x}} = 10$. The four parameters are modeled as Matérn-$3/2$ processes with lengthscales $\ell_{\mathbf{u}_a} = \ell_{\mathbf{u}_b} = \ell_{\mathbf{u}_c} = \ell_{\mathbf{u}_d} = 40$. The diffusion intensities are $\sigma^2_{\mathbf{u}_a} = \sigma^2_{\mathbf{u}_c} = 0.01$ and $\sigma^2_{\mathbf{u}_b} = \sigma^2_{\mathbf{u}_d} = 0.001$. The posterior is computed on a grid from $t_0 = 0$ to $t_{\max} = 60$ units of time with step size $\Delta t = 0.1$.

SIRD As detailed in Section 3.1, the SIRD model is governed by a contact rate $\beta(t)$. Recall that we assume a time-dependent β to account for governmental measures in reaction to the spread of COVID-19. The recovery rate γ and fatality rate η are fixed at $\gamma = 0.06$ and $\eta = 0.002$, like they will be in the experiments with real data in Sections 3.4.2 and 3.4.3 below. As a prior over the dynamics we choose a twice-integrated Wiener process with diffusion intensity $\sigma^2_{\mathbf{x}} = 50$. The contact rate β is modeled as a Matérn-$3/2$ process with lengthscale $\ell_{\mathbf{u}} = 14$ and diffusion intensity $\sigma^2_{\mathbf{u}} = 0.1$. The posterior is computed on a grid from $t_0 = 0$ to $t_{\max} = 100$ units of time with step size $\Delta t = 0.1$.

The model allows for straightforward restriction of parameter values by using link functions. The natural support for the SIRD-contact rate is the interval $[0, 1]$, but $\mathbf{u}(t)$, as a Gauss–Markov process, takes values on the real line. A change in the basis of $\beta(t)$ with a logistic sigmoid function ϑ before it enters the likelihood fixes this misspecification. Similarly, the Lotka-Volterra parameters are inferred in log-space to ensure positivity. It is an appealing aspect of the EKF that these nonlinear transformations do not require significant adaptation of the algorithm. Instead, the EKF treats it as merely another level of linearization of Equation (3.7). Section 3.4.3 extends this to the state dynamics.

The results are shown in Figure 3.2.

On all test problems, the algorithm recovers the true states and the true latent force accurately. The recovery is not exact, which shows how the Gaussian posterior is only an approximation of the true posterior. The χ^2-statistic for the van-der-Pol stiffness

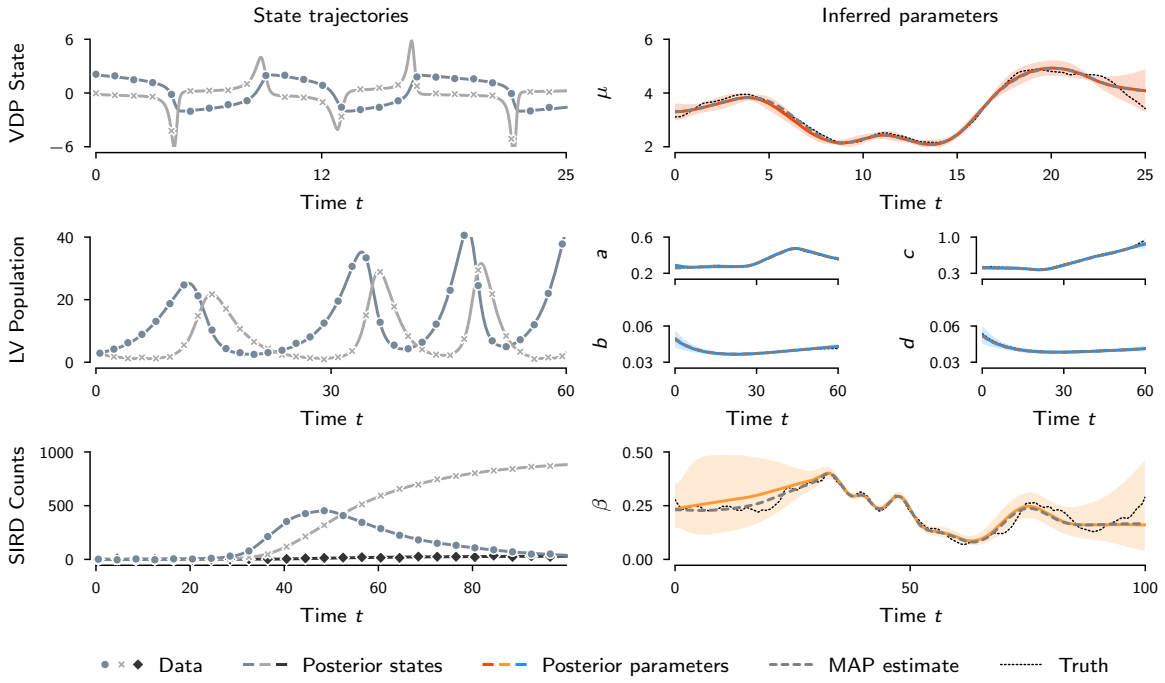


Figure 3.2: **State recovery in simulated environments**. The stiffness parameter of the van-der-Pol oscillator (top row) and the Lotka-Volterra parameters (middle row) are inferred accurately with appropriately high certainty. For the SIRD experiment (bottom row), the uncertainty is high, where low case counts provide little information about the latent contact rate. With more fluctuations in the observed counts, the approximated contact rate displays more certainty.

parameter μ is 1.11, which lies in $(0.0039, 3.8415)$, the 90% confidence interval of the χ^2 distribution with 1 degree of freedom. The root-mean-square error (RMSE) to the truth is 0.14. The χ^2-statistic for the Lotka-Volterra parameters is 8.06, which lies in $(0.7107, 9.4877)$, the 90% confidence interval of the χ^2 distribution with 4 degrees of freedom. The RMSE to the truth is 0.04 in log space and 0.018 in linear space. The χ^2-statistic for the contact rate β is 0.91, which lies in $(0.0039, 3.8415)$, the 90% confidence interval of the χ^2 distribution with 1 degree of freedom. The RMSE to the truth is 0.2 in logit space and 0.033 in linear space.

3.4.2 COVID-19 Data

We continue with the SIRD model introduced in Section 2.1.4, now using data collected in Germany over the period from January 22, 2020, to May 27, 2021. Throughout the pandemic, the German government has imposed mitigation measures of varying severity.

Together with seasonal effects, summer vacations, etc., they caused a continual change in the contact rate. The next experiments aim to recover said contact rate (and the SIRD counts) from the German dataset.

The Center for Systems Science and Engineering at the Johns Hopkins University publishes daily counts of confirmed ($y_n^{\text{confirmed}}$), recovered ($y_n^{\text{recovered}}$), and deceased (y_n^{deceased}) individuals (Dong et al., 2020). One can transform this data to suit the SIRD model

$$I_n := y_n^{\text{confirmed}} - R_n - D_n, \qquad R_n := y_n^{\text{recovered}}, \qquad D_n := y_n^{\text{deceased}}. \tag{3.27}$$

The counts I_n, R_n, and D_n are available for each day, starting with January 22, 2020. Assuming a constant population over time, the numbers of susceptible individuals S_n are always evident from the other quantities, thus left out of the visualizations. We fix the population at $P = 83\,783\,945$, based on public record. We rescale the data to cases per one thousand people (CPT).

As a prior over $\mathbf{x}(t)$, due to its popularity in constructing probabilistic ODE solvers (Tronarp et al., 2019), we assume a twice-integrated Wiener process. $\beta(t)$ is modelled as a Matérn-$3/2$ process with length scale $\ell_q = 75$ and diffusion intensity $\sigma_q^2 = 0.05$. The state-space model is straightforwardly extendable to sums and products of (more) processes (Solin and Särkkä, 2014; Särkkä and Solin, 2019). Inferring parameters that are constant over time, however, is not straightforward due to potentially singular transition models (Särkkä, 2013, Section 12.3.1).

As described in Section 3.4.1, the contact rate is inferred in logit space. We shift the logistic sigmoid function such that it fulfills $\vartheta(0) = 0.1$ in which case the stationary mean $\overline{\mathbf{u}} = 0$ translates to a stationary mean $\vartheta(\overline{\mathbf{u}}) = \overline{\beta} = 0.1$ of the Matérn process that models the contact rate. The recovery rate and mortality rate are considered known and fixed at $\gamma = 0.06$ and $\eta = 0.002$ to isolate the effect of the inference procedure on recovering the evolution of the contact rate $\mathbf{u}(t) = \beta(t)$. We set the mean of the Gaussian initial conditions to the first data point that is available. The diffusion intensity of the prior process $\mathbf{x}(t)$ is set to $\sigma_{\mathbf{x}}^2 = 10$. The latent process $\mathbf{u}$ and all derivatives are initialized at zero. Note that due to the logistic sigmoid transform, an initial value $\mathbf{u}_0 = \mathbf{0}$ amounts to an initial contact rate $\beta_0 = 0.1$.

In the present scenario, we cannot take the SIRD model as an accurate description of the underlying data but merely as a tool that aids the inference engine in recovering physically meaningful states and forces. In order to account for this model mismatch, the Dirac likelihood from Equation (3.7) is relaxed towards a Gaussian likelihood with measurement noise $\lambda^2 = 0.01$. This equals the data observation noise and thus balances the respective impact of either (misspecified) source of information. Intuitively, adding ODE measurement noise reduces how strictly the vector field dynamics are enforced during inference and therefore avoids overconfident estimates of $\beta(t)$.

The mesh-size of the ODE is $\Delta t = 1/24$ days, i.e. ODE updates are computed on an hourly basis. The final 14 observations are excluded from the training set to serve as validation data for evaluating the extrapolation behavior of the proposed method.

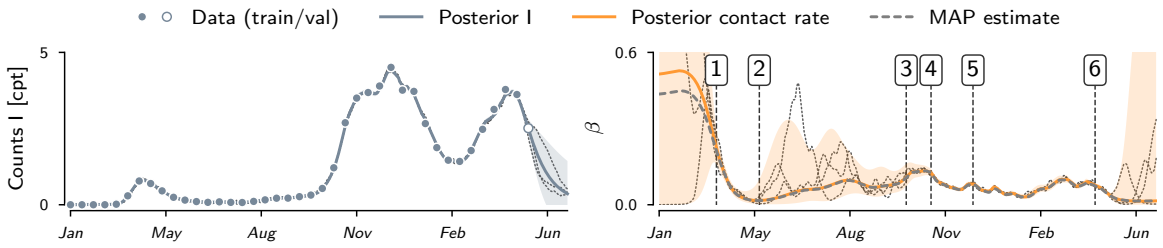


Figure 3.3: **Estimated counts of infectious cases and contact rate based on real COVID-19 data**. The case counts of infectious people are scaled to cases per thousand (cpt). The uncertainty over the contact rate increases when the case counts are low. After a single forward solve, the inferred mean is already close to the MAP estimate. The shaded areas show the 95 % credible interval and the dotted black lines are samples from the posterior.

Table 3.1: List of selected governmental measures imposed in Germany with the aim to contain the spread of COVID-19. These events are depicted in Figures 3.3 and 3.4 (see column 'Mark'). Links to the sources are provided in Appendix A.

Mark	Governmental Measures
1	Stringent contact restrictions, partial shutdown of public life
2 - 3	Continual relaxations of measures
4	Partial shutdown of public life ('lockdown light')
5	Hard lockdown, stringent contact restrictions
6	First nationwide decree of restrictions, intensification of measures

Figure 3.3 shows the results. The mean of the state $\mathbf{x}$ estimates the case counts accurately in both interpolation and extrapolation tasks. The estimated contact rate rapidly decreases around late March, remains low until fall, increases momentarily, and is dampened again soon after. This aligns with a set of political measures imposed by the government (compare Figure 3.3 to Table 3.1). The uncertainty over the estimated contact rate is high in the early beginning when the case counts are still low. It then increases again in summer and with the beginning of the extrapolation phase.

If the experiment is taken as-is, the credibility intervals of the posterior over $\mathbf{x}(t)$ include negative numbers (mostly where the case counts are low and the uncertainty high, and when extrapolating). Of course, in a system that models counts of people in different stages of a disease, negative numbers should be excluded altogether. The proposed method provides straightforward means to address this issue. Section 3.4.3 explains the details.

3.4.3 Nonnegative State Estimates

The following experiment evaluates how the proposed method performs in combination with a state-space model that constrains the support of the dynamics. Concretely, let $\mathbf{x}(t)$ model the logarithm of the SIRD dynamics and the respective derivatives. With a slight abuse of notation, we will continue writing "$\mathbf{x}$" even though it lives in a different space than in the previous sections. The structure of the dynamic model is the same. The diffusion intensity of the prior process $\mathbf{x}(t)$ is $\sigma_{\mathbf{x}}^2 = 0.05$. The diffusion is not comparable to the value in the previous section because the state dynamics moved to log-space. Using $\frac{\mathrm{d}}{\mathrm{d}t}\exp(x(t)) = \exp(x(t))\dot{x}(t)$, the ODE likelihood becomes

$$\mathbf{z}_m \,|\, \mathbf{x}_m^{\text{ODE}}, \mathbf{u}_m^{\text{ODE}}, \sim \mathcal{N}(\boldsymbol{\zeta}_1 - f(\boldsymbol{\zeta}_2;\, \boldsymbol{\zeta}_3), \lambda^2 \boldsymbol{I}_d), \tag{3.28}$$

with auxiliary quantities (recall the logistic sigmoid ϑ)

$$\boldsymbol{\zeta}_1 := \exp\left(\mathbf{x}^{(0)}(t_m^{\text{ODE}})\right) \mathbf{x}^{(1)}(t_m^{\text{ODE}}), \quad \boldsymbol{\zeta}_2 := \exp\left(\mathbf{x}^{(0)}(t_m^{\text{ODE}})\right), \quad \boldsymbol{\zeta}_3 := \vartheta(\mathbf{u}(t_m^{\text{ODE}})). \tag{3.29}$$

The exponential function introduces an additional nonlinearity into the state-space model, which necessitates smaller step-sizes for the ODE measurements (see below).

The observed case count data $\boldsymbol{y}_{0:N}$ is transformed into the log-space, too, in which we assume additive, i.i.d. Gaussian noise. On the one hand, transforming the measurements into log-space implies that the measurement model for the counts remains linear; on the other hand, it imposes a log-normal noise model (if viewed back in "linear space"). Log-normal noise underlines how the estimated states cannot be negative. Again, we scale the counts to cases per thousand.

As depicted in Figure 3.4, the reconstruction of the driving processes in this setting yields results that at first glance, look similar to the previous experiment. The states match the data points well. However, the extrapolation is more realistic in that the credible intervals encode that negative values are impossible (which is due to the log-transform). The mean of the recovered contact rate closely resembles the estimate of the previous experiment. Again, upon implementation of strict governmental measures, the uncertainty decreases, whereas in the context of relaxations, the uncertainty is high.

3.5 Related Work

Latent forces and ODE solvers The explained method closely relates to probabilistic ODE solvers and latent force models (Alvarez et al., 2009), especially the kind of latent force model that exploits the state-space formulation of the prior (Hartikainen et al., 2012). The difference is that, in the spirit of probabilistic numerical algorithms, the mechanistic knowledge in the form of an ODE is injected through the likelihood function instead of the prior. A similar approach of linking observations to mechanistic constraints has previously been used in the literature on constrained Gaussian processes

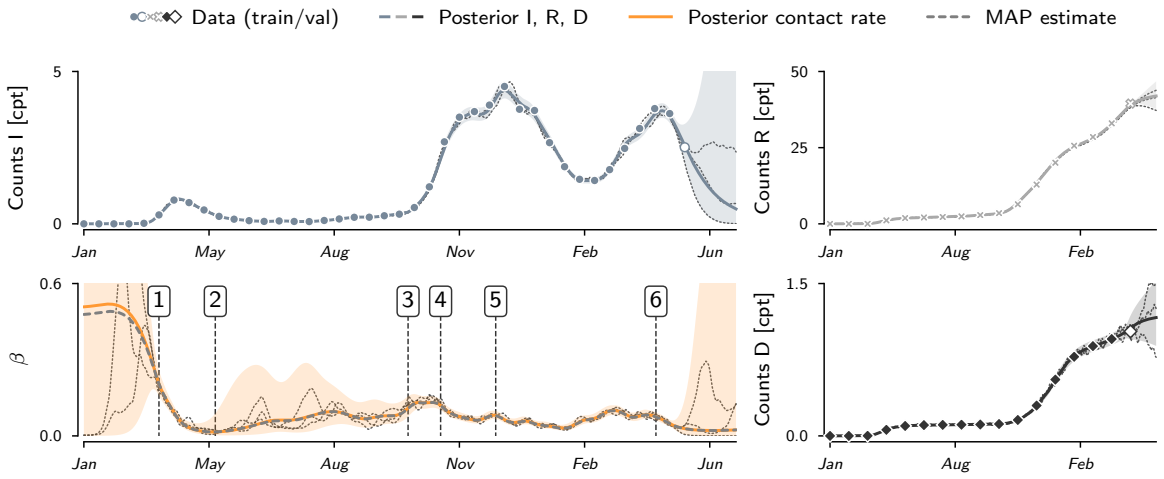


Figure 3.4: **Estimated case counts and contact rate**, inferred in the logarithmic basis on real COVID-19 and vaccination data. The case counts of infectious people are scaled to cases per thousand (cpt). Again, the uncertainty of the contact rate increases where the case counts are low. Now, the posterior credible interval is restricted to the positive reals. The shaded areas show the 95 % credible interval and the dotted black lines are samples from the posterior.

(Jidling et al., 2017) and gradient matching (Calderhead et al., 2009; Wenk et al., 2020). Probabilistic ODE solvers have been used by Kersting et al. (2020a) for efficient ODE inverse problem algorithms, but their approach is different to the present algorithm, in which the need for iterated optimization or sampling is avoided altogether.

Monte Carlo methods (Markov-chain) Monte Carlo methods are also able to infer a time-dependent ODE latent force from a set of state observations. Options that are compatible with a setup similar to the present work would include sequential Monte Carlo techniques (Naesseth et al., 2019), elliptical slice sampling (Murray et al., 2010), or Hamiltonian Monte Carlo (Betancourt, 2017) (for instance realized as the No-U-Turn sampler (Hoffman and Gelman, 2014)). The shared disadvantage of Monte Carlo methods applied to the resulting ODE inverse problem is that the complexity of obtaining *a single* Monte Carlo sample is of the same order of magnitude as computing the *full* Gaussian approximation of the posterior distribution. In Section 3.5.1 we show results from a parametric version of the SIRD-latent force model (using the No-U-Turn sampler as provided by NumPyro (Phan et al., 2019)). This sampler requires *thousands* of numerical ODE solutions, compared to the single solve of our method. This fact is also reflected in the wall-clock time needed for both types of inference. While the MCMC experiment in Section 3.5.1 takes in the order of hours, each experiment with

our approach takes under one minute to complete. In other words, the algorithm in the present work poses an efficient yet expressive alternative to Monte Carlo methods for approximate inference with dynamical systems.

3.5.1 Parametric Model for MCMC Sampling

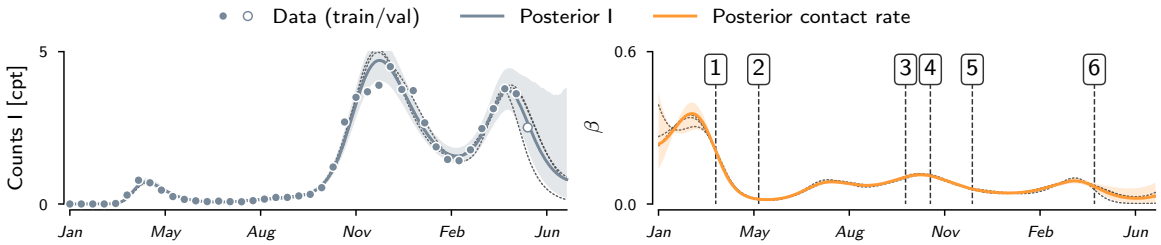


Figure 3.5: **Estimated counts of infectious cases and contact rate**. The estimates are obtained from MCMC sampling in an SIRD model with a parametric function for the contact rate $\beta(t)$. The case counts of infectious people are scaled to cases per thousand (cpt). The shaded areas show the 95 % credible interval and the dotted black lines are samples from the posterior. Compared to the non-parametric approach presented in the paper, the estimate over $\beta(t)$ is very confident in general. The posterior mean closely resembles the results obtained in Sections 3.4.2 and 3.4.3. The numbered markers in the right plot are explained in Table 3.1 in the paper.

This section first introduces a functional form for $\beta(t)$ that connects to the non-parametric model introduced in Section 3.2. Then, a generative model for Markov-chain Monte Carlo (MCMC) inference over the unknown parameters of $\beta(t)$ is set up.

We establish a parametric model for the latent, time-varying contact rate in an SIRD model in terms of Fourier features. In light of Mercer's theorem and the fact that stationary covariance functions have complex-exponential eigenfunctions (Rasmussen and Williams, 2006, Chapter 4.3), this closely connects to the Matérn-$3/2$ process used in Sections 3.4.2 and 3.4.3 (see also (Rahimi and Recht, 2008)).

Concretely, we proceed as follows. Let $\mathbb{T}$ denote a dense time grid. First, (i) compute the kernel Gram matrix $\boldsymbol{K}$ on $\mathbb{T}$, such that $K_{ij} = k(x_i, x_j)$ with $x_i, x_j \in \mathbb{T}$. k is the Matérn-$3/2$ covariance function. As in the experiments before, we set the characteristic lengthscale to $\ell = 75$. Then, (ii) compute the eigendecomposition of $\boldsymbol{K}$. In order to keep the dimensionality of the inference problem feasible, select $r \ll |\mathbb{T}|$ eigenvectors that correspond to the r largest eigenvalues of $\boldsymbol{K}$. In this experiment, we choose $r = 25$. (iii) For each eigenvector, the strongest frequency component ω is determined by the discrete Fourier decomposition. This yields a set of frequencies $\{\omega_i : i = 1, \ldots, r\}$. Finally, the

parametric model is defined as the sum of parametrized Fourier features of the form

$$\beta(t) = \vartheta\left(\sum_{i=1}^{r} a_i \cos\left(2\pi\omega_i t\right) + b_i \sin\left(2\pi\omega_i t\right)\right), \tag{3.30}$$

where ϑ is the logistic sigmoid function as described in Section 3.4. We aim to compute a posterior contact rate $\beta(t)$ by MCMC inference over the coefficients a_i and b_i, $i = 1, \ldots, r$. To this end, we define a prior over the parameter vector $\boldsymbol{\theta} := (a_1, b_1, \ldots, a_r, b_r)^\top$ and a likelihood for the COVID-19 case counts $\boldsymbol{y}_{0:N}$ with respect to $\boldsymbol{\theta}$.

In order to ensure non-negative case counts, as in Section 3.4.3, we assume log-normally distributed measurements with i.i.d. noise

$$p(\mathbf{y}_{0:N} \mid \boldsymbol{\theta}) = \prod_{n=0}^{N} \text{LogNormal}\left(\mathbf{y}_n; \log\left(\boldsymbol{x}^{(\theta)}(t_n)\right), \sigma^2 \boldsymbol{I}_{2r}\right), \tag{3.31}$$

where σ^2 is inferred from the data along with $\boldsymbol{\theta}$. $\boldsymbol{x}^{(\boldsymbol{\theta})}(t_n)$ denotes the solution of the SIRD system at time t_n, parametrized by the vector of coefficients $\boldsymbol{\theta}$ through the contact rate from Equation (3.30). Notably, each evaluation of the likelihood involves numerically integrating the SIRD system, which significantly increases the computational cost entailed by the inference algorithm. This is done by NumPyro's DOPRI-5 implementation (Phan et al., 2019; Dormand and Prince, 1980).

The prior distributions over the Fourier-feature coefficients and over σ^2 are chosen as

$$p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\right), \qquad p(\sigma^2) = \text{HalfCauchy}(\sigma^2; 0.01). \tag{3.32}$$

The mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ of the prior over $\boldsymbol{\theta}$ is set to a maximum-likelihood estimate by minimizing the negative logarithm of Equation (3.31) with SciPy's L-BFGS optimization algorithm (Virtanen et al., 2020; Liu and Nocedal, 1989). The covariance is chosen as $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = 0.1 \cdot \boldsymbol{I}_{2r}$.

The goal of the experiment is to compute a posterior over the coefficients $\boldsymbol{\theta}$ (and the measurement covariance σ^2) that is comparable to the results obtained in Sections 3.4.2 and 3.4.3. Like before, recovery rate and fatality rate are assumed fixed and known at $\gamma = 0.06$ and $\eta = 0.002$. We compute the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{y}_{0:N})$ using NumPyro's implementation of the No-U-Turn sampler (Hoffman and Gelman, 2014).

Figure 3.5 shows the estimated number of infectious people and the contact rate over time as inferred by the MCMC algorithm. The state estimate matches the data points well and the uncertainty increases when extrapolating. Like in the experiments in Sections 3.4.2 and 3.4.3, the final 14 observations serve as a validation set and the model extrapolates 31 days into the future. The posterior mean closely resembles the results obtained from our method. However, the uncertainty is lower in general, especially in the beginning and over the summer months.

4 Probabilistic Numerical Method of Lines

So far, the dynamics formalized solely a temporal unfolding of the state vector. By considering partial differential equations, we avail ourselves of the possibility to describe the interaction of spatially related state components.

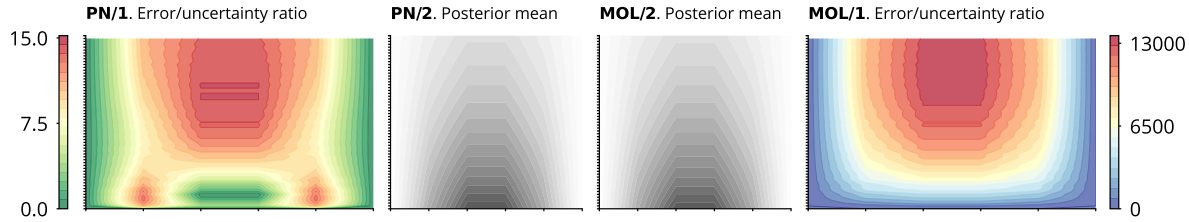


Figure 4.1: **PNMOL fixes bad calibration of the MOL/ODE filter combination:** Posterior means and error/uncertainty ratios of PNMOL (left) and MOL with a conventional PN solver (right) on a fine time grid (y-axis) and a coarse space grid (x-axis) for the heat equation. The means are indistinguishable (*/2). PN w/ MOL is poorly calibrated (error/uncertainty ratios $\sim 10^5$; */1), but PNMOL acknowledges all inaccuracies.

Recall the setup from Section 2.5, in which we want to approximate an unknown function $\boldsymbol{u}$ that solves

$$\frac{\partial}{\partial t}\boldsymbol{u}(t,x) = F(t,x,\boldsymbol{u}(t,x),\mathcal{D}\boldsymbol{u}(t,x)), \tag{4.1}$$

for $t \in [t_0, t_{\max}]$ and $x \in \Omega$, subject to initial condition

$$\boldsymbol{u}(t_0,x) = h(x), \quad x \in \Omega, \tag{4.2}$$

and boundary conditions $\mathcal{B}\boldsymbol{u}(t,x) = g(x)$, $x \in \partial\Omega$. In Section 2.5.1, we introduced the method of lines, a technique that makes using numerical solving algorithms for ODEs possible in order to solve PDEs. This approach has one central problem. Discretizing the spatial domain, and only then applying an ODE solver, turns the PDE solver into a *pipeline of two numerical algorithms* instead of a single algorithm. This is bad because as a result of this serialisation, the error estimates returned by the ODE solver are unreliable. The solver lacks crucial information about whether the spatial grid consists of, say, $N = 4$ or $N = 10^7$ points. As will be confirmed by the experiments in Section 4.6, it

is intuitive that a coarse spatial grid puts a lower bound on the overall precision, even if the ODE solver uses small time steps. But since this is not "known" to the ODE solver, not even to a probabilistic one, it may waste computational resources by needlessly decreasing its step-size and may deliver (severely) overconfident uncertainty estimates (for example, Figure 4.1). Such overconfident uncertainty estimates are essentially useless to a practitioner, which complicates the usage of method of lines approaches within probabilistic simulation of differential equations; for example, in the context of latent force inference or inverse problems.

The present chapter provides a solution to this problem by revealing a way of tracking spatiotemporal correlations in the PDE solutions while preserving the computational advantages of traditional MOL through a probabilistic numerical solver. The error that is introduced by approximating the differential operator $\mathcal{D}$ with a matrix $\boldsymbol{D}$ can be quantified probabilistically, and its time-evolution acts as a latent force on the resulting ODE. The combined posterior over the PDE solution and the latent force can be computed efficiently with a PN ODE solver. More specifically, we contribute:

- ◇ *Probabilistic discretization:* Sections 4.1 and 4.2 present a probabilistic technique for constructing finite-dimensional approximations of $\mathcal{D}$ that quantify the discretization error. It translates an unsymmetric collocation methods (Kansa, 1990) to the language of Gaussian process interpolation and admits efficient, sparse approximations akin to radial-basis-function-generated finite-difference formulas (Tolstykh and Shirobokov, 2003).

- ◇ *Calibrated PDE filter:* Building on the probabilistic discretization, Section 4.3 explains how the discretization and the ODE solver need not be treated as two separate black-box entities. By applying the idea behind the method of lines to a *global* Gaussian process prior, it becomes evident how the discretization uncertainty naturally evolves over time. With a filtering-based probabilistic ODE solver ("ODE filter", c.f. Section 2.4), posteriors over the latent error process and the PDE solution can be inferred in linear time, and without disregarding spatial uncertainties like traditional MOL algorithms do.

Section 4.5 explains hyperparameter choices and calibration. Section 4.7 discusses connections to existing work. Section 4.6 demonstrates the advantages of the approach over conventional MOL. Altogether, PNMOL enriches the toolbox of probabilistic simulations of differential equations by a calibrated and efficient PDE solver.

Note regarding notation This chapter will use non-boldface notation (e.g. x) for sets or vectors containing spatial grid points, even though the major part of this work uses boldface notation for vectors. This is to avoid notational confusion with the state $\boldsymbol{x}$ of an ODE. The unknown function is here denoted $\boldsymbol{u}$, instead of $\boldsymbol{x}$.

4.1 PN Discretization

This section derives how to approximate a differential operator $\mathcal{D}$ by a matrix $\boldsymbol{D}$. Let $\mathbf{u}_x(z) \sim \mathcal{GP}(0, \gamma^2 k_x)$ be a Gaussian process on Ω with covariance kernel $k_x(z, z')$ and output scale $\gamma > 0$. The presentation in Section 4.3 will assume a Gaussian process prior $\mathbf{u}(t, x) \sim \mathcal{GP}(0, \gamma^2 k_t \otimes k_x)$, thus the notation "$\mathbf{u}_x$" motivates $\mathbf{u}_x$ as "the x-part of $\mathbf{u}(t, x)$".

The objective is to approximate the PDE dynamics in a way that circumvents the differential operator $\mathcal{D}$,

$$f(t, \mathbb{X}, \mathbf{u}_x(\mathbb{X})) \approx F(t, \mathbb{X}, \mathbf{u}_x(\mathbb{X}), (\mathcal{D}\mathbf{u}_x)(\mathbb{X})), \tag{4.3}$$

that is, $\mathcal{D}\mathbf{u}_x$ disappears from Equation (4.1) because f replaces F. For linear differential operators $\mathcal{D}$, this reduces to replacing $(\mathcal{D}\mathbf{u}_x)(\mathbb{X})$ with a matrix-vector product $\boldsymbol{D}\mathbf{u}_x(\mathbb{X})$, $\boldsymbol{D} \in \mathbb{R}^{(N+1)\times(N+1)}$ (recall Equation (2.127)). $\mathcal{D}$ must not be present in the discretized model, because otherwise, the PDE does not translate into a system of ODEs, and we cannot proceed with the method of lines. But with an approximate derivative based on only matrix-vector arithmetic, the computational efficiency of ODE solvers can be exploited to infer an approximate PDE solution.

Since $\mathbf{u}$ is a Gaussian process, applying the linear operator $\mathcal{D}$ yields another Gaussian process $\mathcal{D}\mathbf{u}_x \sim \mathcal{GP}(0, \mathcal{D}^2 k_x)$, where we abbreviate $\mathcal{D}^2 k_x(z, z') = \mathcal{D}\mathcal{D}^* k_x(z, z')$ and $\mathcal{D}^*$ is the adjoint of $\mathcal{D}$ (in the present context, this means applying $\mathcal{D}$ to z' instead of z). Conditioning $\mathbf{u}_x$ on realisations of $\mathbf{u}_x(\mathbb{X})$, and then applying $\mathcal{D}$, yields another Gaussian process with moments

$$\hat{\boldsymbol{m}}(z) = \boldsymbol{W}_{\mathbb{X}}(z)\mathbf{u}_x(\mathbb{X}), \tag{4.4a}$$

$$\hat{\boldsymbol{K}}(z, z') = \gamma^2 \left[(\mathcal{D}^2 k_x)(z, z') - \boldsymbol{W}_{\mathbb{X}}(z) k_x(\mathbb{X}, \mathbb{X}) \boldsymbol{W}_{\mathbb{X}}(z')^\top\right], \tag{4.4b}$$

$$\boldsymbol{W}_{\mathbb{X}}(z) := (\mathcal{D}k_x)(z, \mathbb{X}) k_x(\mathbb{X}, \mathbb{X})^{-1}. \tag{4.4c}$$

Let $\boldsymbol{\xi}_x \sim \mathcal{GP}(0, \gamma^2 \hat{\boldsymbol{K}})$. $\boldsymbol{\xi}_x$ is independent of $\mathbf{u}_x$, since $\hat{\boldsymbol{K}}$ only depends on $\mathcal{D}$, k_x, and $\mathbb{X}$, not on $\mathbf{u}_x(\mathbb{X})$. Then,

$$\boldsymbol{W}_{\mathbb{X}}(z)\mathbf{u}_x(\mathbb{X}) + \boldsymbol{\xi}_x(z) \sim \mathcal{GP}(0, \mathcal{D}\mathcal{D}^* k_x) \tag{4.5}$$

follows, which implies

$$p(\mathcal{D}\mathbf{u}_x(\cdot)) = p(\boldsymbol{W}_{\mathbb{X}}(\cdot)\mathbf{u}_x(\mathbb{X}) + \boldsymbol{\xi}(\cdot)). \tag{4.6}$$

In particular, when evaluated on the grid, we obtain the matrix-vector formulation of the differential operator

$$(\mathcal{D}\mathbf{u}_x)(\mathbb{X}) = \boldsymbol{W}_{\mathbb{X}}(\mathbb{X})\mathbf{u}_x(\mathbb{X}) + \boldsymbol{\xi}_x(\mathbb{X}). \tag{4.7}$$

Equation (4.7) yields $\mathcal{D}\mathbf{u}_x$ solely as a transformation of values of $\mathbf{u}_x$ with known, additive Gaussian noise $\boldsymbol{\xi}_x$. Altogether, the above derivation identifies the *differentiation matrix* $\boldsymbol{D}$ and the *error covariance* $\boldsymbol{E}$

$$\boldsymbol{D} := \boldsymbol{W}_{\mathbb{X}}(\mathbb{X}), \quad \boldsymbol{E} := \hat{\boldsymbol{K}}(\mathbb{X}, \mathbb{X}). \tag{4.8}$$

If we know $\mathbf{u}_x(\mathbb{X})$, we have an approximation to $(\mathcal{D}\mathbf{u}_x)(\mathbb{X})$ that is wrong with covariance $\gamma^2\boldsymbol{E}$. The matrix $\boldsymbol{E}$ makes the approach probabilistic (and new).

There are two possible interpretations for $\boldsymbol{D}$ (and $\boldsymbol{E}$). $\boldsymbol{D}$ appears in a method for solving PDEs $\mathcal{D}\mathbf{u}_x = f$ with radial basis functions, called *unsymmetric collocation*, or *Kansa's method* (Kansa, 1990; Hon and Schaback, 2001; Schaback, 2007). A related method, called symmetric collocation (Fasshauer, 1997, 1999), has been built on and translated into a Bayesian probabilistic numerical method by Cockayne et al. (2017). The following derivation shows that a translation is possible for the probabilistic discretization, respectively unsymmetric collocation, as well: Equation (4.6) explains

$$\begin{aligned}(\mathcal{D}\mathbf{u}_x)(\mathbb{X}) \mid \mathbf{u}_x(\mathbb{X}), \boldsymbol{\xi}_x &\sim \delta\left[D\mathbf{u}_x(\mathbb{X}) + \boldsymbol{\xi}_x\right] && \text{(4.9a)}\\ &= \mathcal{N}(D\mathbf{u}_x(\mathbb{X}), \gamma^2\boldsymbol{E}). && \text{(4.9b)}\end{aligned}$$

In other words, $\boldsymbol{D}$ and $\boldsymbol{E}$ (respectively $\boldsymbol{D}$ and $\boldsymbol{\xi}$) estimate $\mathcal{D}\mathbf{u}_x$ from values of $\mathbf{u}_x$ at a set of grid-points – just like finite difference formulas do. By construction, it is a Bayesian probabilistic numerical algorithm. Cockayne et al. (2019) introduce a formal definition of Bayesian and non-Bayesian probabilistic numerical methods. More specifically, they define a probabilistic numerical method to be a tuple of an *information operator* and a *belief update operator*. A PN method then becomes Bayesian if its output is the pushforward of a specific conditional distribution through the *quantity of interest*. All of those objects can be derived for the PN discretization of $\mathcal{D}$. The same is true for the boundary conditions explained in Section 4.4 but is omitted in the following. The proof of the statement below mirrors the explanation why Bayesian quadrature is a Bayesian PN method in Section 2.2 of the paper by Cockayne et al. (2019).

Proposition 10. *The approximation of* $(\mathcal{D}\mathbf{u}_x)(\mathbb{X})$ *with* $\boldsymbol{D}$ *and* $\boldsymbol{E}$ *as in Equations* (4.6) *and* (4.8)*, using evaluations of* $\mathbf{u}_x$ *at a grid, is a Bayesian probabilistic numerical method.*

Proof. The prior measure is the GP prior $\mathcal{GP}(0, \gamma^2 k_x)$ and defined over some separable Banach space of sufficiently differentiable, real-valued functions. The information operator $\mathcal{I}[\varphi] := \varphi(\mathbb{X})$ evaluates a function φ at the grid $\mathbb{X}$. The belief update restricts the prior measure to the set of functions that interpolate $\varphi(\mathbb{X})$ (this is standard Gaussian process conditioning). The quantity of interest is the derivative $\mathcal{D}\mathbf{u}_x$, which results in the posterior distribution in Equation (4.4). □

The limitations of the above probabilistic discretization are twofold. First, computation of the differentiation matrix $\boldsymbol{D}$ involves inverting the kernel Gram matrix $k_x(\mathbb{X}, \mathbb{X})$.

For large point sets, this matrix is notoriously ill-conditioned (Schaback, 1995). Second, computation – and application – of $\boldsymbol{D}$ is expensive because $\boldsymbol{D}$ is a dense matrix with as many rows and columns as there are points in $\mathbb{X}$. Since loosely speaking, a derivative is a "local" phenomenon (other than e.g. computing integrals), intuition suggests that $\mathcal{D}\mathbf{u}_x$ can be approximated more cheaply by exploiting this locality. This thought leads to localized differentiation matrices and probabilistic numerical finite differences.

4.2 PN Finite Differences

The central idea of an efficient approximation of the probabilistic discretization is to approximate the derivative of $\mathbf{u}_x$ at each point x_n in $\mathbb{X}$ individually. This leads to a row-by-row assembly of $\boldsymbol{D}$, which can be sparsified naturally and without losing the ability to quantify numerical discretization/differentiation uncertainty.

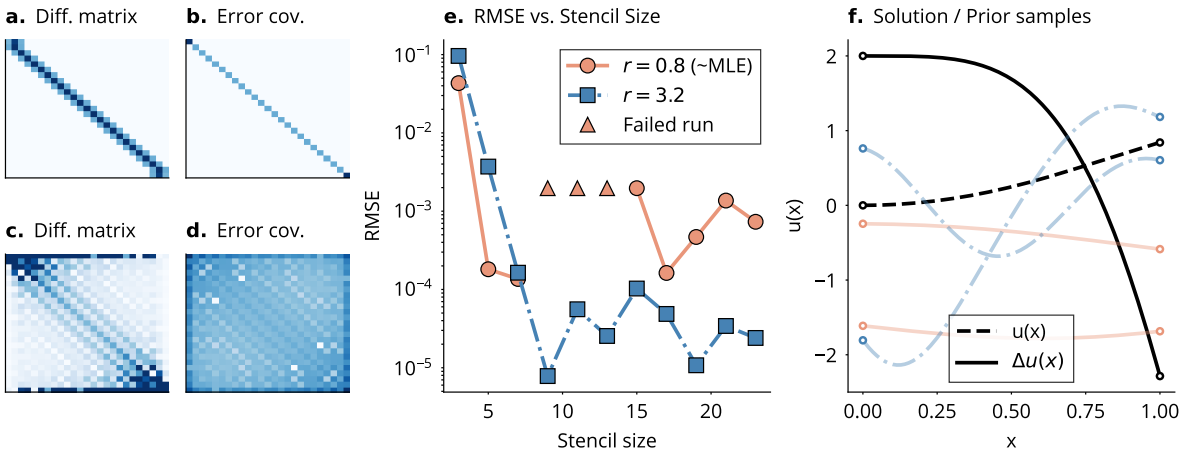


Figure 4.2: **Discretize the Laplacian with a local and global approximation:** The target is the Laplacian $\mathcal{D} = \Delta$ of $u(x) = \sin(\|x\|^2)$ (f). *Left:* Sparsity pattern of the differentiation matrix and error covariance matrix for the localized approximation (a, b) and the global approximation (c, d) on a mesh with $N = 25$ points. The approximation is least certain at the boundaries. *Centre:* The root-mean-square error between Δu and its approximation decreases with an increased stencil size but the approximation breaks down for larger stencils (e), likely due to ill-conditioned kernel Gram matrices. A maximum likelihood estimate of the input scale $r \in \mathbb{R}$ of the square exponential kernel $k(x, y) = e^{-r^2\|x-y\|^2}$ based on data $u_x(x)$ alone does not necessarily lead to well-conditioned system matrices, nor does it generally imply a low RMSE (e). *Right:* Samples from the prior GP u_x for both length scales are shown next to the solution and the target function (f; the colours match the colours in the RMSE plot). *Increasing stencil sizes improves the accuracy until stability concerns arise.*

Write the matrix $\boldsymbol{D}$ as a vertical stack of $N + 1$ row-vectors, $\boldsymbol{D} = (\boldsymbol{d}_0, ..., \boldsymbol{d}_N)^\top$, and denote by e_n the n-th diagonal element of $\boldsymbol{E}$, i.e. the variance of the approximation of

$\mathcal{D}\mathbf{u}_x$ at the n-th grid-point in $\mathbb{X}$. Thus,

$$\mathcal{D}\mathbf{u}_x(x_n) \mid \mathbf{u}_x(\mathbb{X}), \boldsymbol{\xi}_x \sim \delta[\boldsymbol{d}_n^\top \mathbf{u}_x(\mathbb{X}) + \xi_{x,n}] \tag{4.10}$$

with the scalar random variable $\xi_{x,n} \sim \mathcal{N}(0, \gamma^2 e_n)$. If, in Equation (4.10), we replace $\mathbf{u}_x(\mathbb{X})$ by the values of $\mathbf{u}_x$ at only a local neighbourhood of x_n (a "stencil"),

$$x_{\mathrm{loc}(n)} = \{x_{n-k}, ..., x_n, ..., x_{n+k}\}, \tag{4.11}$$

the general form of Equation (4.10) is preserved, but $\boldsymbol{d}_n$ consists of only $2k+1$ elements, with $k \ll N$,

$$\boldsymbol{d}_n = (\mathcal{D}k_x)(x_n, x_{\mathrm{loc}(n)}) k_x(x_{\mathrm{loc}(n)}, x_{\mathrm{loc}(n)})^{-1} \tag{4.12}$$

instead of $N+1$ elements. The error e_n becomes

$$e_n := (\mathcal{D}^2 k_x)(x_n, x_n) - \boldsymbol{d}_n k_x(x_{\mathrm{loc}(n)}, x_{\mathrm{loc}(n)}) \boldsymbol{d}_n^\top. \tag{4.13}$$

If the coefficients in the new $\boldsymbol{d}_n$ are all embedded into $\boldsymbol{D}$ according to the indices of $x_{\mathrm{loc}(n)}$ in $\mathbb{X}$, $\boldsymbol{D}$ becomes sparse and $\boldsymbol{E}$ becomes diagonal. More precisely, $D_{ij} \neq 0$ if and only if $j \in \mathrm{loc}(i)$. On a one-dimensional domain Ω, and with $k = 1$, $\boldsymbol{D}$ is a banded matrix with bandwidth 3. Furthermore, due to the point-by-point construction, choosing $k = N/2$ implies that the original $\boldsymbol{D}$ from Section 4.1 is recovered; however, $\boldsymbol{E}$ remains diagonal. The sparse approximation resolves many of the performance issues that persist with the global approximation (Figure 4.2).

The sparsified differentiation matrix is known as the radial-basis-function-generated finite difference matrix (Driscoll and Fornberg, 2002; Shu et al., 2003; Tolstykh and Shirobokov, 2003; Fornberg and Flyer, 2015). The correspondence to finite-difference formulas stems from the fact that if k_x were a polynomial kernel and the mesh $\mathbb{X}$ were equidistant and one-dimensional, the coefficients in $\boldsymbol{D}$ would equal the standard finite difference coefficients (Fornberg, 1988). The advantage of the more general, kernel-based finite difference approximation over polynomial coefficients is a more robust approximation, especially for non-uniform grid points and in higher dimensions (Tolstykh and Shirobokov, 2003; Fornberg and Flyer, 2015). The Bayesian point of view does not only add uncertainty quantification in the form of $\boldsymbol{E}$ but also reveals that a practitioner may choose suitable kernels k_x to include prior information into the PDE simulation.

4.3 PN Method of Lines

The previous two sections have been concerned with approximating the derivative of a function, purely from observations of this function on a grid. Next, we use these strategies to solve time-dependent PDEs. To this end, we combine the probabilistic dis-

cretization with an ODE filter. As opposed to non-probabilistic MOL, this combination quantifies the leak of information between the space discretization and the ODE solution. Simply put, what is commonly treated as a pipeline of disjoint solvers, becomes more of a single algorithm.

4.3.1 Spatiotemporal Prior Process

For any function $\varphi = \varphi(t)$, let $\bar{\boldsymbol{\varphi}}$ be the stack of φ and its first ν derivatives, $\bar{\boldsymbol{\varphi}}(t) := (\varphi(t), \dot{\varphi}(t), ..., \varphi^{(\nu)}(t))$. This abbreviation is convenient because some stochastic processes (like the integrated Wiener or Matérn process) do not have the Markov property, but the stack of state and derivatives does. The following assumption is integral for the probabilistic method of lines.

Assumption 4.1. *Assume a Gaussian process prior with separable covariance structure,*

$$\mathbf{u} = \mathbf{u}(t, x) \sim \mathcal{GP}(0, \gamma^2 k_t \otimes k_x) \tag{4.14}$$

for some output-scale $\gamma > 0$.

Compared to the traditional method of lines, where the temporal and spatial dimensions are treated independently and with black-box methods, Assumption 4.1 is mild: albeit the covariance is separable, the algorithm still starts with a single, global Gaussian process. Assumption 4.1 allows choosing temporal kernels that eventually lead to a fast algorithm:

Assumption 4.2. *Assume k_t is a covariance kernel that allows conversion to a linear, time-invariant stochastic differential equation in the following sense: For any $\boldsymbol{\upsilon} \sim \mathcal{GP}(0, \gamma^2 k_t)$, $\bar{\boldsymbol{\upsilon}}$ solves the SDE*

$$\mathrm{d}\bar{\boldsymbol{\upsilon}}(t) = \boldsymbol{A}\bar{\boldsymbol{\upsilon}}(t)\,\mathrm{d}t + \boldsymbol{B}\,\mathrm{d}\mathbf{w}(t) \tag{4.15}$$

subject to Gaussian initial conditions

$$\bar{\boldsymbol{\upsilon}}(t_0) \sim \mathcal{N}(\boldsymbol{m}_0, \gamma^2 \boldsymbol{C}_0), \tag{4.16}$$

for $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{m}_0$, and $\boldsymbol{C}_0$ that derive from k_t, and for a one-dimensional Wiener process $\mathbf{w}$ with diffusion γ^2.

Assumption 4.2 is satisfied, for instance, for the integrated Wiener process or the Matérn process; many more examples are given in Chapter 12 of the book by Särkkä and Solin (2019). Assumptions 4.1 and 4.2 unlock the machinery of probabilistic ODE solvers.

Next, we add spatial correlations into the prior SDE model. The following Lemma 11 will simplify the derivations further below (Solin, 2016).

Lemma 11. *Let k_t be a covariance function that satisfies Assumption 4.2. Let $\boldsymbol{M} \in \mathbb{R}^{q\times q}$ be a matrix. For a process $\boldsymbol{\varphi} \sim \mathcal{GP}(0, \gamma^2 k_t \otimes \boldsymbol{M})$, $\bar{\boldsymbol{\varphi}}$ solves*

$$\mathrm{d}\bar{\boldsymbol{\varphi}}(t) = (\boldsymbol{A} \otimes \boldsymbol{I})\bar{\boldsymbol{\varphi}}(t)\,\mathrm{d}t + (\boldsymbol{B} \otimes \boldsymbol{I})\,\mathrm{d}\hat{\mathbf{w}}(t), \tag{4.17}$$

subject to Gaussian initial conditions

$$\bar{\boldsymbol{\varphi}}(t_0) \sim \mathcal{N}(\boldsymbol{m}_0 \otimes \mathbb{1}, \gamma^2 \boldsymbol{C}_0 \otimes \boldsymbol{M}), \tag{4.18}$$

where $\hat{\mathbf{w}}$ is a q-dimensional Wiener process with constant diffusion $\gamma^2\boldsymbol{M}$, and $\boldsymbol{I} \in \mathbb{R}^{q\times q}$ is the identity. $\boldsymbol{m}_0 \otimes \mathbb{1}$ is a vertical stack of d copies of $\boldsymbol{m}_0$.

Lemma 11 suggests that a spatiotemporal prior (Assumption 4.1) may be restricted to a spatial grid without losing the computational benefits that SDE priors provide. Abbreviate $\mathbf{U}(t) := \mathbf{u}(t, \mathbb{X})$. Lemma 11 gives rise to an SDE representation for $\mathbf{U}(t)$ by choosing $\boldsymbol{M} = k_x(\mathbb{X}, \mathbb{X})$. The same holds for the error model: Recall the definition of the error covariance $\boldsymbol{E}$ from Equation (4.8) (or Equation (4.13) respectively, if the localized version is used). $\boldsymbol{\xi} \sim \mathcal{GP}(0, \gamma^2 k_t \otimes \boldsymbol{E})$ admits a state-space formulation due to Lemma 11. Through

$$\begin{aligned}
&\boldsymbol{D}\mathbf{u}(t,\mathbb{X}) + \boldsymbol{\xi}(t) && \text{(4.19a)}\\
&\quad\sim \mathcal{GP}(0, \gamma^2 k_t \otimes [\boldsymbol{D}k_x(\mathbb{X},\mathbb{X})\boldsymbol{D}^\top + \boldsymbol{E}]) && \text{(4.19b)}\\
&\quad= \mathcal{GP}(0, \gamma^2 k_t \otimes (\mathcal{D}\mathcal{D}^* k_x)(\mathbb{X},\mathbb{X})]) && \text{(4.19c)}
\end{aligned}$$

it is evident that $\boldsymbol{\xi}$ is an appropriate prior model for the time-evolution of the spatial discretization error. This error being part of the probabilistic model is the advantage of PNMOL over non-probabilistic versions.

4.3.2 Information Model

The priors over $\mathbf{U}$ and $\boldsymbol{\xi}$ become a probabilistic numerical PDE solution by conditioning $\mathbf{U}$ and $\boldsymbol{\xi}$ on "solving the PDE" at a number of grid points as follows. Recall from Equation (4.7) how $\mathcal{D}\mathbf{u}(t, \mathbb{X})$ is approximated by $\boldsymbol{D}\mathbf{u}(t, \mathbb{X}) + \boldsymbol{\xi}$. The residual process

$$\mathbf{r}(t) := \dot{\mathbf{U}}(t) - F(t, \mathbb{X}, \mathbf{U}(t), \boldsymbol{D}\mathbf{U}(t) + \boldsymbol{\xi}) \tag{4.20}$$

measures how well realisations of $\bar{\mathbf{U}}$ and $\bar{\boldsymbol{\xi}}$ solve the PDE. $\dot{\mathbf{U}}$ and $\mathbf{U}$ are components in $\bar{\mathbf{U}}$, therefore all operations are tractable given the extended state vectors $\bar{\mathbf{U}}$ and $\bar{\boldsymbol{\xi}}$. Conditioning $\bar{\mathbf{U}}$ and $\bar{\boldsymbol{\xi}}$ on $\mathbf{r}(t) = \mathbf{0}$ for all t yields a probabilistic PDE solution. However, in practice, we need to discretize the time variable first in order to be able to compute the (approximate) posterior.

4.3.3 Time-Discretisation

Let $\mathbb{T} := (t_0, ..., t_K)$ be a grid on $[t_0, t_{\max}]$. Define the time-steps $h_k := t_{k+1} - t_k$. Restricted to the time grid,

$$\bar{\mathbf{U}}(t_{k+1}) \mid \bar{\mathbf{U}}(t_k) \sim \mathcal{N}(\boldsymbol{\Phi}(h_k)\bar{\mathbf{U}}(t_k), \gamma^2 \boldsymbol{Q}_{\mathbf{U}}(h_k)) \tag{4.21a}$$

$$\bar{\boldsymbol{\xi}}(t_{k+1}) \mid \bar{\boldsymbol{\xi}}(t_k) \sim \mathcal{N}(\boldsymbol{\Phi}(h_k)\bar{\boldsymbol{\xi}}(t_k), \gamma^2 \boldsymbol{Q}_{\boldsymbol{\xi}}(h_k)) \tag{4.21b}$$

with transition matrix

$$\boldsymbol{\Phi}(h_k) := \breve{\boldsymbol{\Phi}}(h_k) \otimes \boldsymbol{I} := \exp(\boldsymbol{A}h_k) \otimes \boldsymbol{I} \tag{4.22a}$$

and process noise covariance matrices

$$\boldsymbol{Q}_{\mathbf{U}}(h_k) := \breve{\boldsymbol{Q}}(h_k) \otimes k_x(\mathbb{X}, \mathbb{X}) \tag{4.23a}$$

$$\boldsymbol{Q}_{\boldsymbol{\xi}}(h_k) := \breve{\boldsymbol{Q}}(h_k) \otimes \boldsymbol{E} \tag{4.23b}$$

$$\breve{\boldsymbol{Q}}(h_k) := \int_0^{h_k} \breve{\boldsymbol{\Phi}}(h_k - \tau)\boldsymbol{B}\boldsymbol{B}^\top \breve{\boldsymbol{\Phi}}(h_k - \tau)^\top \, \mathrm{d}\tau. \tag{4.23c}$$

$\breve{\Phi}$ and $\breve{\boldsymbol{Q}}$ can be computed efficiently with matrix fractions. Integrated Wiener process priors, their time-discretizations, and practical considerations for implementation of high-orders are discussed in the probabilistic ODE solver literature (e.g. Tronarp et al., 2021; Krämer and Hennig, 2020). On the discrete-time grid, the information model reads

$$\mathbf{r}(t_k) \mid \bar{\mathbf{U}}(t_k), \bar{\boldsymbol{\xi}}(t_k) \sim \delta[\dot{\mathbf{U}}(t_k) - F(t, \mathbb{X}, \mathbf{U}(t_k), \boldsymbol{D}\mathbf{U}(t_k) + \boldsymbol{\xi}(t_k))]. \tag{4.24}$$

The complete setup is depicted in Figure 4.3.

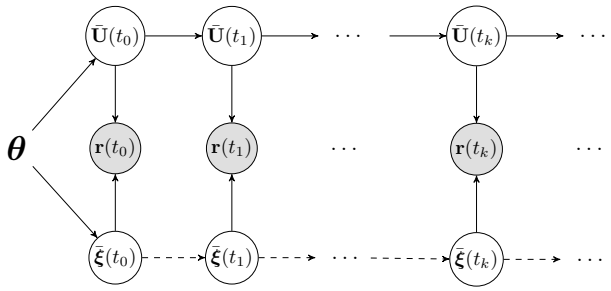


Figure 4.3: **Graphical visualisation of PNMOL:** The states $\bar{\boldsymbol{\xi}}$ and $\bar{\mathbf{U}}$ are conditionally independent given $\boldsymbol{\theta} = (k_x, k_t, \mathbb{X}, \mathcal{D}, \gamma, \mathbb{T})$. *The existence of the bottom row distinguishes PNMOL from other PDE solvers.* The dependencies between the $\boldsymbol{\xi}(t_k)$, indicated by the dashed lines, are optional; removing them improves the efficiency of the algorithm; details are in Section 4.3.5.

4.3.4 Inference

Altogether, the probabilistic method of lines targets

$$p\left(\bar{\mathbf{U}}, \bar{\boldsymbol{\xi}} \;\middle|\; [\mathbf{r}(t_k) = \mathbf{0}]_{k=0}^{K}\right) \tag{4.25}$$

which we will refer to as the *probabilistic numerical PDE solution* of Equation (4.1). The precise form of the PN PDE solution is intractable because F (and therefore the likelihood in Equation (4.24)) are nonlinear. It can be approximated efficiently with techniques from extended Kalman filtering (Särkkä and Solin, 2019), which is based on approximating the nonlinear PDE vector field F with a first-order Taylor series. Inference in the linearized model is feasible with Kalman filtering and smoothing. Särkkä and Solin (2019) summarize the details, and Tronarp et al. (2019) explain specific considerations for ODE solvers.

The role of $\boldsymbol{\xi}$ and its impact on the posterior distribution (i.e. the PDE solution) is comparable to that of a latent force in the ODE (Alvarez et al., 2009; Hartikainen et al., 2012; Schmidt et al., 2021). The coarser $\mathbb{X}$ is, the larger is $\boldsymbol{E}$, respectively $\boldsymbol{\xi}$. A large $\boldsymbol{\xi}$ indicates how increasingly severe the misspecification of the ODE becomes. Unlike traditional PDE solvers, inference according to the information model in Equation (4.24) embraces latent discretization errors. It does so by an algorithm that shares similarities with the latent force inference algorithm by Schmidt et al. (2021) (compare Figure 4.3 to Schmidt et al. (2021, Fig. 3)), but the sources of misspecification are different in both works.

Inference of $\boldsymbol{\xi}$ is expensive because the complexity of a Gaussian filter scales as $O(Kd^3)$ where d is the dimension of the state-space model. In the present case, $d = 2N(\nu + 1)$ because for N spatial grid points, PNMOL tracks ν time-derivatives of $\mathbf{U}$ and $\boldsymbol{\xi}$. Both $\mathbf{U}$ and $\boldsymbol{\xi}$ are $(N+1)$-dimensional. If the only purpose of $\boldsymbol{\xi}$ is to incorporate a measure of spatial discretization uncertainty into the information model of the PDE solver, tracking $\boldsymbol{\xi}$ in the state space is not required, as long as we introduce another approximation.

4.3.5 White-Noise Approximation

Recall $\boldsymbol{\xi} \sim \mathcal{GP}(0, \gamma^2 k_t \otimes \boldsymbol{E})$, i.e. $\boldsymbol{\xi}$ is an integrated Wiener process in time and a Gaussian $\mathcal{N}(0, \gamma^2 \boldsymbol{E})$ in space. One may relax the temporal integrated Wiener process prior to a white noise process prior; that is, $\boldsymbol{\xi}(t)$ is independent of $\boldsymbol{\xi}(s)$ for $s \neq t$, and $\boldsymbol{\xi}(t) \sim \mathcal{N}(0, \gamma^2 \boldsymbol{E})$ for all t. The state-space realisation of a white noise process is trivial because there is no temporal evolution (recall the dashed lines in Figure 4.3). To understand the impact of $\boldsymbol{E}$ in the white-noise formulation on the information model, consider the PDE vector field $F(t, x, \boldsymbol{u}(t, x), \mathcal{D}\boldsymbol{u}(t, x))$. The central idea is to regard F as a function of $\mathbf{U}$ and $\boldsymbol{\xi}$, instead of $\boldsymbol{u}$ and $\mathcal{D}\boldsymbol{u}$, and linearize each nonlinearity with a first-order Taylor series. The Dirac likelihood in Equation (4.24) then becomes a Gaussian with the measurement noise $\boldsymbol{E}$. This section derives this for linear and nonlinear PDEs.

Linear PDEs

At first, we explain the general idea for a linear PDE

$$F(t, x, \boldsymbol{u}(t,x), \mathcal{D}\boldsymbol{u}(t,x)) = \boldsymbol{A}\boldsymbol{u}(t,x) + \boldsymbol{B}\mathcal{D}\boldsymbol{u}(t,x) \tag{4.26}$$

for some coefficients $\boldsymbol{A}$ and $\boldsymbol{B}$. If $\boldsymbol{u}$ is scalar-valued, $\boldsymbol{A}$ and $\boldsymbol{B}$ are scalars. If $\boldsymbol{u}$ is vector-valued, $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices. We use bold-faced notation for the latter, more general case. $\mathcal{D}\mathbf{u} = \boldsymbol{D}\mathbf{u} + \boldsymbol{\xi}$ allows for recasting the vector field F as a function of $\mathbf{u}$ and $\boldsymbol{\xi}$, instead of $\boldsymbol{u}$ and $\mathcal{D}\boldsymbol{u}$,

$$\tilde{F}(t, \mathbf{U}(t), \boldsymbol{\xi}(t)) := F(t, \mathbb{X}, \mathbf{U}(t), \boldsymbol{\xi}(t)) = \boldsymbol{A}\mathbf{U}(t) + \boldsymbol{B}\boldsymbol{D}\mathbf{U}(t) + \boldsymbol{B}\boldsymbol{\xi}(t). \tag{4.27}$$

Since this equation is linear in $\boldsymbol{u}$ and $\mathcal{D}\boldsymbol{u}$, thus also in $\mathbf{U}$ and $\boldsymbol{\xi}$, there is no need for Taylor-series approximation. The linear measurement model emerges as

$$\mathbf{r}_{\text{linear}}(t_k) \mid \bar{\mathbf{U}}(t_k), \bar{\boldsymbol{\xi}}(t_k) \sim \delta\left[\dot{\mathbf{U}}(t_k) - \boldsymbol{A}\mathbf{U}(t_k) - \boldsymbol{B}\boldsymbol{D}\mathbf{U}(t_k) - \boldsymbol{B}\boldsymbol{\xi}(t_k)\right]. \tag{4.28}$$

If the temporal evolution of $\boldsymbol{\xi}$ is ignored through replacing k_t with a white-noise approximation as in Section 4.3.5, Equation (4.28) becomes

$$\mathbf{r}_{\text{linear}}(t_k) \mid \bar{\mathbf{U}}(t_k) \sim \mathcal{N}\left(\dot{\mathbf{U}}(t_k) - \boldsymbol{A}\mathbf{U}(t_k) - \boldsymbol{B}\boldsymbol{D}\mathbf{U}(t_k), \boldsymbol{B}\boldsymbol{E}\boldsymbol{B}^\top\right). \tag{4.29}$$

The same principle can be generalized to fully nonlinear problems as follows.

Nonlinear PDEs

Next, consider a fully nonlinear PDE vector field $F(t, x, \boldsymbol{u}(t,x), \mathcal{D}\boldsymbol{u}(t,x))$. This general setting includes semilinear and quasilinear systems of equations. As before, we discretize the spatial domain using $\mathbb{X}$, use $\mathcal{D}\mathbf{u} = \boldsymbol{D}\mathbf{U} + \boldsymbol{\xi}$, rewrite

$$\tilde{F}(t, \mathbf{U}(t), \boldsymbol{\xi}(t)) := F(t, \mathbb{X}, \mathbf{u}(t, \mathbb{X}), \boldsymbol{D}\mathbf{u}(t, \mathbb{X}) + \boldsymbol{\xi}(t)), \tag{4.30}$$

and infer the solution via $\tilde{F}$. Since $\tilde{F}$ is nonlinear, we have to linearize it before proceeding. Let $\nabla_{\mathbf{U}}\tilde{F}$ be the derivative of $\tilde{F}$ with respect to $\mathbf{U}$, and $\nabla_{\boldsymbol{\xi}}\tilde{F}$ be its $\boldsymbol{\xi}$-counterpart. We linearize $\tilde{F}$ at some $\boldsymbol{\eta} = (\boldsymbol{\eta}_{\mathbf{U}}, \boldsymbol{\eta}_{\boldsymbol{\xi}}) \in \mathbb{R}^{2(N+1)}$,

$$\tilde{F}(t, \mathbf{U}(t), \boldsymbol{\xi}(t)) \approx \tilde{F}(t, \boldsymbol{\eta}_{\mathbf{U}}, \boldsymbol{\eta}_{\boldsymbol{\xi}}) + \nabla_{\mathbf{U}}\tilde{F}(t, \boldsymbol{\eta}_{\mathbf{U}}, \boldsymbol{\eta}_{\boldsymbol{\xi}})(\mathbf{U}(t) - \boldsymbol{\eta}_{\mathbf{U}}) + \nabla_{\boldsymbol{\xi}}\tilde{F}(t, \boldsymbol{\eta}_{\mathbf{U}}, \boldsymbol{\eta}_{\boldsymbol{\xi}})(\xi(t) - \boldsymbol{\eta}_{\boldsymbol{\xi}}) \tag{4.31}$$

$$=: \boldsymbol{H}_{\mathbf{U}}\mathbf{U}(t) + \boldsymbol{H}_{\boldsymbol{\xi}}\boldsymbol{\xi}(t) + \boldsymbol{b}(t). \tag{4.32}$$

$\boldsymbol{\eta}_{\mathbf{U}}$ and $\boldsymbol{\eta}_{\boldsymbol{\xi}}$ are commonly chosen as the predicted mean of $\boldsymbol{\xi}$ and $\mathbf{U}$, which corresponds to extended Kalman filtering (Särkkä and Solin, 2019). As a result of the linearization,

the information model reads

$$\mathbf{r}_{\text{nonlinear}}(t_k) \mid \bar{\mathbf{U}}(t_k), \bar{\boldsymbol{\xi}}(t_k) \sim \delta\left[\dot{\mathbf{U}}(t_k) - \boldsymbol{H}_{\mathbf{U}}\mathbf{U}(t_k) - \boldsymbol{H}_{\boldsymbol{\xi}}\boldsymbol{\xi}(t_k) - \boldsymbol{b}\right], \tag{4.33}$$

or, for the white-noise version,

$$\mathbf{r}_{\text{nonlinear}}(t_k) \mid \bar{\mathbf{U}}(t_k) \sim \mathcal{N}\left(\dot{\mathbf{U}}(t_k) - \boldsymbol{H}_{\mathbf{U}}\mathbf{U}(t_k) - \boldsymbol{b}, \boldsymbol{H}_{\boldsymbol{\xi}}\boldsymbol{E}\boldsymbol{H}_{\boldsymbol{\xi}}^{\top}\right). \tag{4.34}$$

Equations (4.33) and (4.34) are both linear in the states, and inference is possible with Kalman filtering and smoothing.

Remark 12. *In many practical applications, semi-linear PDEs are of special interest. Concretely,*

$$F_{semi}(t, x, u, \mathcal{D}u) := \mathcal{D}u(t, x) + f(u(t, x)), \tag{4.35}$$

for some nonlinear f and an additional spatial diffusion defined via $\mathcal{D}$. In this special case of nonlinear PDEs, Equation (4.24) *becomes*

$$\mathbf{r}_{white}(t_k) \mid \bar{\mathbf{U}}(t_k) \sim \mathcal{N}(\rho(\bar{\mathbf{U}}(t)), \gamma^2\boldsymbol{E}), \tag{4.36a}$$

$$\rho(\bar{\mathbf{U}}(t)) := \dot{\mathbf{U}}(t) - \boldsymbol{D}\mathbf{U}(t) - f(\mathbf{U}(t)). \tag{4.36b}$$

The advantage of the white-noise approximation over the latent-force version is that the state-space model is precisely half the size because $\boldsymbol{\xi}$ is not a state variable anymore. Since the complexity of PNMOL depends cubically on the dimension of the state-space, half as many state variables improve the complexity of the algorithm by a factor $2^3 = 8$. Arguably, $\gamma^2\boldsymbol{E}$ entering the information model as a measurement covariance may also provide a more intuitive explanation of the impact that the statistical quantification of the discretization error has on the PDE solution: the larger $\boldsymbol{E}$, the less strictly is a zero PDE residual enforced during inference. The error covariance $\boldsymbol{E}$ regularizes the impact of the information model, and, in the limit of $\boldsymbol{E} \to \infty$, yields the prior as a PDE solution. Intuitively put, for $\boldsymbol{E} > \mathbf{0}$, the algorithm puts less trust in the residual information $\mathbf{r}(\cdot)$ than for $\boldsymbol{E} = \mathbf{0}$.

4.4 Boundary Conditions

So far, the existence of the boundary operator $\mathcal{B}$ was neglected, since it can be treated in the same way as $\mathcal{D}$. In order to complete the derivation, this section provides complete formulas, including $\mathcal{B}$.

4.4.1 Discretized Boundary Conditions

To augment the PDE residual information with boundary conditions, we begin by discretizing $\mathcal{B}$ probabilistically; either with global collocation as in Section 4.1 or with PN finite differences as in Section 4.2. Below, we show the former, because the latter becomes accessible with the same modifications from Section 4.2. Let $\mathbf{u}_x \sim \mathcal{GP}(0, \gamma^2 k_x)$. Defining the differentiation matrix $\boldsymbol{B}$ and the error covariance $\boldsymbol{R}$,

$$\boldsymbol{B} := (\mathcal{B}k_x)(\mathbb{X}, \mathbb{X})k_x(\mathbb{X}, \mathbb{X})^{-1}, \quad \boldsymbol{R} := (\mathcal{B}^2 k_x)(\mathbb{X}, \mathbb{X}) - \boldsymbol{B}k_x(\mathbb{X}, \mathbb{X})\boldsymbol{B}^\top, \tag{4.37}$$

we have access to boundary conditions: For any $\boldsymbol{y} \in \mathbb{R}^{N+1}$,

$$p((\mathcal{B}\mathbf{u}_x)(\mathbb{X}) \mid \mathbf{u}_x(\mathbb{X}) = \boldsymbol{y}) \sim \mathcal{N}(\boldsymbol{B}\boldsymbol{y}, \gamma^2 \boldsymbol{R}) \tag{4.38}$$

holds. For Dirichlet boundary conditions, $\mathcal{B}$ is the identity, thus $\boldsymbol{B} = \boldsymbol{I}_{N+1}$ is the identity matrix and $\boldsymbol{R} \equiv \mathbf{0}$ is the zero matrix. For Neumann conditions, $\boldsymbol{B}$ is the discretized derivative along normal coordinates, and $\boldsymbol{R}$ is generally nonzero (recall the explanation in Section 4.5 of the cases in which the error matrix is zero). The derivation surrounding Equation (4.38) above suggests how the present framework would deal with boundary conditions that are subject to additive Gaussian noise.

4.4.2 Latent Force

Define the latent force $\boldsymbol{\vartheta} = \boldsymbol{\vartheta}(t) \sim \mathcal{GP}(0, \gamma^2 k_t \otimes \boldsymbol{R})$, which will play a role similar to $\boldsymbol{\xi}$ but for the boundary conditions. $\boldsymbol{\vartheta}$ inherits a stochastic differential equation formulation from k_t just like $\boldsymbol{\xi}$ does (recall Lemma 11). Denote the stack of $\boldsymbol{\vartheta}$ and its first $\nu \in \mathbb{N}$ time-derivatives by $\bar{\boldsymbol{\vartheta}}$. Let $\mathbb{X}_B \subset \mathbb{X}$ be the subset of boundary points in $\mathbb{X}$. The full information operator (i.e. an extended version of Equation (4.20)) includes $\mathcal{B}\boldsymbol{u}(t, x)|_{\partial\Omega} = g(x)$ as

$$\mathbf{r}_{\text{full}}(t) := \begin{pmatrix} \dot{\mathbf{U}}(t) - F(t, \mathbb{X}, \mathbf{U}(t), \boldsymbol{D}\mathbf{U}(t) + \boldsymbol{\xi}) \\ \mathcal{B}\mathbf{U}(t) - g(\mathbb{X}_B) - \boldsymbol{\vartheta}(t) \end{pmatrix} \overset{!}{=} \mathbf{0}. \tag{4.39}$$

$\mathbf{r}_{\text{full}}$ depends on $\bar{\mathbf{U}}$, $\bar{\boldsymbol{\xi}}$, and $\bar{\boldsymbol{\vartheta}}$. Conditioning $(\bar{\mathbf{U}}, \bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\vartheta}})$ on $\mathbf{r}_{\text{full}}(t) = \mathbf{0}$ yields a probabilistic PDE solution that knows boundary conditions. Notably, the bottom row in $\mathbf{r}_{\text{full}}$ is linear in $\left(\bar{\mathbf{U}}, \bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\vartheta}}\right)$ so there is no linearization required to enable (approximate) inference.

4.4.3 Comparison to MOL

Boundary conditions in PNMOL enter through the information operator, i.e. on the same level as the PDE vector field F, which is different to conventional MOL: In MOL, one only tracks the state variables in the interior of Ω, i.e. $u(t, \mathbb{X}\backslash\mathbb{X}_B)$ because boundary conditions can be inferred from the interior straightforwardly. For Dirichlet conditions,

the boundary values are always dictated by $g(\mathbb{X}_B)$. Let $\frac{\partial}{\partial n}$ be the directional derivative taken in the direction normal to the boundary $\partial\Omega$. For Neumann conditions, for some small $\lambda > 0$, we can approximate

$$\frac{\partial}{\partial n} u(t,x) = \frac{u(t,x) - u(t, x - \lambda n)}{\lambda} \overset{!}{=} g(x). \tag{4.40}$$

This viewpoint suggests $u(t,x) = \lambda g(x) + u(t, x - \lambda n)$ and λ can be chosen such that $x - \lambda n$ is the nearest neighbor of $x \in \mathbb{X}$. While tracking only the state values in the interior of the domain has the advantage that the ODE system emerging from the method of lines is smaller than for PNMOL (which perhaps explains why in Figure 4.4, PNMOL achieves a lower error than MOL), MOL has two disadvantages: (i) non-deterministic boundary conditions are not straightforward to include; (ii) the finite difference approximation of Neumann conditions introduces errors. PNMOL does not face the first issue and quantifies the error mentioned by the second issue.

4.5 Hyperparameters

4.5.1 Kernels

As common in the literature on probabilistic ODE solvers, we use temporal integrated Wiener process priors (e.g. Tronarp et al., 2019; Bosch et al., 2021). In the experiments, the order of integration is $\nu \in \{1, 2\}$. Using low order ODE solvers is not unusual for MOL implementations (Cash and Psihoyios, 1996). Spatial kernels need to be sufficiently differentiable to admit the formula in Equation (4.8). In this work, we use squared exponential kernels. Choosing their input scale is not straightforward (recall Figure 4.2; we found $r = 0.25$ to work well across experiments). Other sufficiently regular kernels, e.g. rational quadratic or Matérn kernels, would work as well. Polynomial kernels recover traditional finite difference weights (Fornberg, 1988), but like for any other feature-based kernel, $k(x,y) = \boldsymbol{\Phi}(x)^\top \boldsymbol{\Phi}(y)$ holds, thus both summands in Equation (4.8) (and in Equation (4.13)) cancel out. The discretization uncertainty $\boldsymbol{E}$ would be zero. In this case, PNMOL gains similarity to the traditional method of lines combined with a probabilistic ODE solver. (The boundary conditions would be treated slightly differently; we refer to Section 4.4.)

4.5.2 Spatial Grid

The spatial grid can be any set of freely scattered points. Spatial neighbourhoods can, for example, be queried from a KD tree (Bentley and Ottmann, 1979). For the simulations in the present paper, we use equispaced grids. PNMOL's requirements on the grid differ from, for instance, finite element methods, in that there needs to be no notion of

connectivity or triangulation between the grid points. PN finite differences only require stencils; in light of the stability results in Figure 4.2, we choose them maximally small.

4.5.3 Output Scale

The output-scale γ calibrates the width of the posterior and can be tuned with quasi-maximum likelihood estimation. Omitting the boundary conditions, this means (recall $\mathbf{r}$ from Equation (4.24))

$$(\hat{\gamma})^2 := \frac{1}{(N+1)(K+1)} \sum_{k=0}^{K} \|\mathbb{E}[\mathbf{r}(t_k)]\|^2_{\mathbb{C}[\mathbf{r}(t_k)]^{-1}}, \tag{4.41}$$

where the Mahalanobis norm $\|\boldsymbol{x}\|_{\boldsymbol{A}} = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}$ is used.

The mean $\mathbb{E}[\mathbf{r}(t_k)]$ and the covariance between the output-dimensions of $\mathbf{r}(\cdot)$ at time t_k, $\mathbb{C}[\mathbf{r}(t_k)]$, emerge from the same Gaussian approximation that computes the approximate posterior. Proving the validity of the estimator in Equation (4.41) parallels similar statements for similar settings (Tronarp et al., 2019; Bosch et al., 2021; Krämer and Hennig, 2021) and consists of two phases: (i) showing that the posterior covariances are of the form $\boldsymbol{C}_k = \gamma^2 \breve{\boldsymbol{C}}_k$ for some $\breve{\boldsymbol{C}}_k$, $k = 0, ..., K$; (ii) deriving the maximum likelihood estimators.

To show the first claim, recall the discrete-time transition of $\mathbf{U}$ and $\boldsymbol{\xi}$ from Equation (4.21) and the information model from Equation (4.24). Then, the fully discretized state-space model is

$$\bar{\mathbf{U}}(t_0) \sim \mathcal{N}(\boldsymbol{m}_0 \otimes \mathbb{1}, \gamma^2 \boldsymbol{C}_0 \otimes k_x(\mathbb{X}, \mathbb{X})) \tag{4.42a}$$

$$\bar{\boldsymbol{\xi}}(t_0) \sim \mathcal{N}(\boldsymbol{m}_0 \otimes \mathbb{1}, \gamma^2 \boldsymbol{C}_0 \otimes \boldsymbol{E}) \tag{4.42b}$$

$$\bar{\mathbf{U}}(t_{k+1}) \mid \bar{\mathbf{U}}(t_k) \sim \mathcal{N}(\boldsymbol{\Phi}(h_k)\bar{\mathbf{U}}(t_k), \gamma^2 \boldsymbol{Q}_{\mathbf{U}}(h_k)) \tag{4.42c}$$

$$\bar{\boldsymbol{\xi}}(t_{k+1}) \mid \bar{\boldsymbol{\xi}}(t_k) \sim \mathcal{N}(\boldsymbol{\Phi}(h_k)\bar{\boldsymbol{\xi}}(t_k), \gamma^2 \boldsymbol{Q}_{\boldsymbol{\xi}}(h_k)) \tag{4.42d}$$

$$\mathbf{r}(t_k) \mid \bar{\mathbf{U}}(t_k), \bar{\boldsymbol{\xi}}(t_k) \sim \delta\left(\dot{\mathbf{U}}(t_k) - F(t, \mathbb{X}, \mathbf{U}(t_k), \boldsymbol{D}\mathbf{U}(t_k) + \boldsymbol{\xi}(t_k))\right). \tag{4.42e}$$

All transitions have a process noise that depends multiplicatively on γ^2. The Dirac likelihood is noise-free. Therefore, the posterior covariance depends multiplicatively on γ as well (which can be proved by an induction that is identical to the one in Appendix C of the paper by Tronarp et al. (2019)).

To show the second claim, let a linearized observation model be given by

$$\mathbf{r}(t_k) \mid \bar{\mathbf{U}}(t_k), \bar{\boldsymbol{\xi}}(t_k) \sim \delta\left(\boldsymbol{H}\begin{pmatrix}\bar{\mathbf{U}}(t_k) \\ \bar{\boldsymbol{\xi}}(t_k)\end{pmatrix} + \boldsymbol{b}\right) \tag{4.43}$$

for appropriate $\boldsymbol{H}$ and $\boldsymbol{b}$ (which are explained in Section 4.3.5). Now, the distribution $p(\mathbf{r}(t_k) \mid \mathbf{r}(t_{k-1}))$ is Gaussian and thus fully defined by its mean $\mathbb{E}[\mathbf{r}(t_k)]$ and its covariance

$\mathbb{C}[\mathbf{r}(t_k)]$. The covariance $\mathbb{C}[\mathbf{r}(t_k)]$ is again of the form $\gamma^2 \breve{\boldsymbol{S}}_k$ for some $\breve{\boldsymbol{S}}_k$ because every filtering covariance is, and the information operator is noise-free. Due to the prediction error decomposition (Schweppe, 1965),

$$p(\mathbf{r}(t_0), ..., \mathbf{r}(t_K) \mid \gamma) = p(\mathbf{r}(t_0) \mid \gamma) \prod_{k=1}^{N} p(\mathbf{r}(t_k) \mid \mathbf{r}(t_{k-1}), \gamma). \tag{4.44}$$

Recall that the dimension of $\mathbf{r}(t_k)$ is $N+1$ because $\mathbb{X}$ consists of $N+1$ grid points. Because everything is Gaussian, the negative log-likelihood (as a function of x) decomposes into

$$-\log p(x; \mathbf{r}(t_0), ..., \mathbf{r}(t_K) \mid \gamma) = \frac{1}{2} \left(\sum_{k=0}^{K} \|x - \mathbb{E}[\mathbf{r}(t_k)]\|^2_{\mathbb{C}[\mathbf{r}(t_k)]^{-1}} - (K+1)(N+1) \log \gamma^2 \right). \tag{4.45}$$

Setting the γ-derivative of the negative log-likelihood to zero, i.e. maximizing it with respect to γ, yields the MLE from Equation (4.41). Overall, the derivation is very similar to those provided by Tronarp et al. (2019); Bosch et al. (2021) for ODE initial value problems, and Krämer and Hennig (2021) for ODE boundary value problems.

4.6 Experiments

Quantify the global error We investigate how the PN method of lines impacts numerical uncertainty quantification. As a first experiment, we solve a spatial Lotka-Volterra model (Holmes et al., 1994), i.e. nonlinear predator-prey dynamics with spatial diffusion, on a range of temporal and spatial resolutions. From the results in Figure 4.4, it is evident how the spatial accuracy limits the overall accuracy. But also how traditional ODE filters combined with MOL fail to quantify numerical uncertainty reliably. At any parameter configuration, it is *either* the spatial or the temporal discretization that dominates the error. Decreasing the time-step alone lets the error not only stagnate but worsens the calibration because the ODE solver does not know how bad the spatial approximation is.

Which error dominates? To further examine which one of *either* Δx or Δt dominates the approximation, we consider a second example: a spatial SIR model (Example 9). We investigate more formally how increasing either, the time-resolution vs. the space-resolution, leads to a low overall error. The results are in Figure 4.5, and confirm the findings from Figure 4.4 above. Traditional PN ODE solvers with conventional MOL are unaware of the true, global approximation error. PNMOL is not, despite being equally accurate.

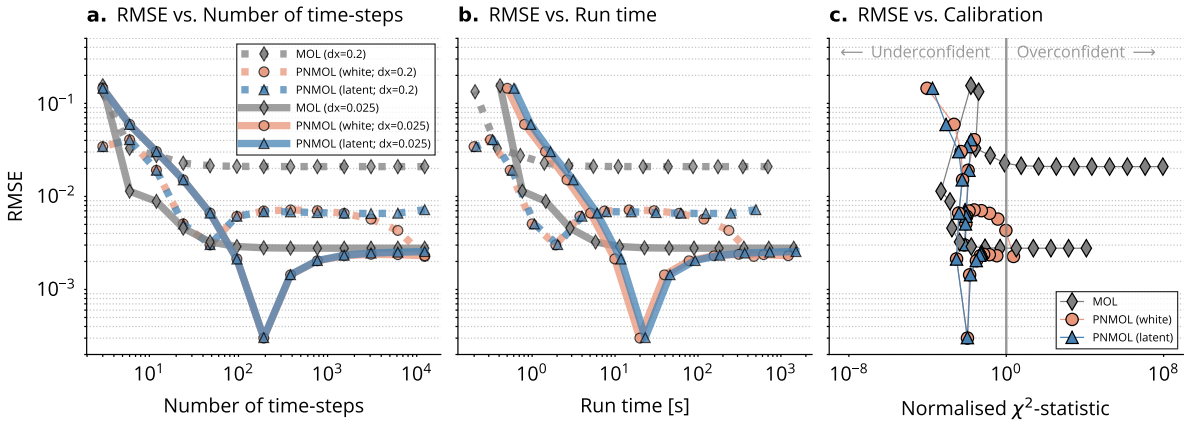


Figure 4.4: **Quantify the global error:** Work vs. precision vs. calibration of PNMOL in the latent-force version (blue) and the white-noise version (orange), compared to a traditional PN ODE solver combined with conventional MOL (grey), on the spatial Lotka-Volterra model. Two kinds of curves are shown: one for a coarse (dotted), and one for a fine spatial mesh (solid). A reference is computed by discretizing the spatial domain with a ten times finer mesh and solving the ODE with backward differentiation formulas. The RMSE of both methods stagnates once a certain accuracy is reached, but PNMOL appears to reach a slightly lower RMSE for $\Delta x = 0.2$ (left, middle; perhaps due to different treatment of boundary conditions; Section 4.4). The run time of PNMOL-white is comparable to that of MOL, and the run time of PNMOL-latent is slightly longer (middle). The calibration of PNMOL, measured in the normalized χ^2-statistic of the Gaussian posterior (so that the "optimum" is 1, not d), remains close to 1 but is slightly underconfident. With decreasing time-steps, MOL is poorly calibrated.

4.7 Related Work

Connections to non-probabilistic numerical approximation have been discussed in Sections 4.1 and 4.2. Recall from there that the strongest connections are to unsymmetric collocation (Kansa, 1990; Hon and Schaback, 2001; Schaback, 2007), radial-basis-function-generated finite differences (Driscoll and Fornberg, 2002; Shu et al., 2003; Tolstykh and Shirobokov, 2003; Fornberg and Flyer, 2015), and collocation methods in general. We refer to Fasshauer (2007); Fornberg and Flyer (2015) for a more comprehensive overview. The literature on the method of lines is covered by, e.g., Schiesser (2012). Dereli and Schaback (2013); Hon et al. (2014) combine collocation with MOL. None of the above exploits the correlations between spatial and temporal errors. The significance of estimating the interplay of both error sources for MOL has been recognized by Berzins (1988); Lawson et al. (1991); Berzins et al. (1991).

Cockayne et al. (2017); Owhadi (2015, 2017); Raissi et al. (2017, 2018) describe a probabilistic solver for PDEs relating to symmetric collocation approaches from numerical analysis. Chen et al. (2021) extend the ideas to nonlinear PDEs. Wang et al.

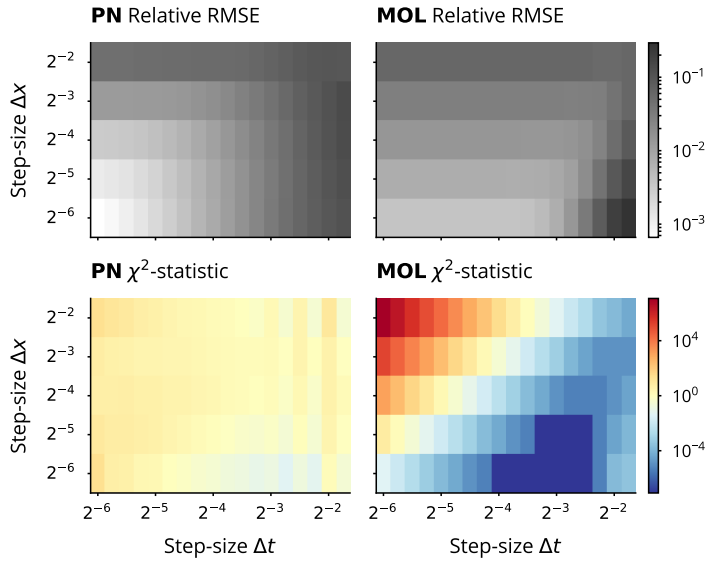


Figure 4.5: **Which error dominates?** The relative RMSE is only small if *both* Δt and Δx are small, which affects all solvers: PNMOL (top left) as well as traditional MOL combined with PN ODE solvers (top right). PN w/ MOL is severely overconfident for large Δx and small Δt (bottom right), while PNMOL delivers a calibrated posterior distribution (bottom left).

(2021) continue the work of Chkrebtii et al. (2016) in constructing an ODE/PDE initial value problem solver that uses (approximate) conjugate Gaussian updating at each time-step. Duffin et al. (2021) solve time-dependent PDEs by discretizing the spatial domain with finite elements, and applying ensemble and extended Kalman filtering in time. They build on the paper by Girolami et al. (2021). Conrad et al. (2017); Abdulle and Garegnani (2021) compute probabilistic PDE solutions by randomly perturbing non-probabilistic solvers. All of the above discard the uncertainty associated with discretizing $\mathcal{D} \approx D$. Some papers achieve ODE-solver-like complexity for time-dependent problems (Wang et al., 2021; Chkrebtii et al., 2016; Duffin et al., 2021), while others compute a continuous-time posterior (Cockayne et al., 2017; Owhadi, 2015, 2017; Raissi et al., 2017, 2018; Chen et al., 2021). PNMOL does both.

The efficiency of the PDE filter builds on recent work on filtering-based probabilistic ODE solvers (Schober et al., 2019a; Tronarp et al., 2019; Kersting et al., 2020b; Kersting and Hennig, 2016; Bosch et al., 2021; Krämer and Hennig, 2020; Tronarp et al., 2021) and their applications (Kersting et al., 2020a; Schmidt et al., 2021). Frank and Enßlin (2020) apply an ODE filter to solve discretized PDEs. Similar algorithms have been developed for other types of ODEs (Hennig and Hauberg, 2014; John et al., 2019; Krämer and Hennig, 2021). The papers by Chkrebtii et al. (2016); Conrad et al. (2017); Abdulle and Garegnani (2021), prominently feature ODEs.

5 Probabilistic ODE Solutions in Extremely High Dimensions

In Chapter 4 we learned how to phrase a PDE as an ODE in order to avail ourselves of efficient ODE solvers. From a practical perspective, we thereby neglected the fact that the discretized differential operator results in extremely large matrices – depending on the chosen grid. Concretely, the matrix $\boldsymbol{D} \approx \mathcal{D}$ will have $\prod_{n=0}^{N} d_n$ entries for N dimensions, and d_n grid points in dimension n. As detailed in Chapter 4, it is crucial for well-calibrated probabilistic PDE solutions that we do not neglect the error arising from discretizing $\mathcal{D}$. Therefore, it is necessary to make probabilistic ODE sovlers fast and ready for extremely large systems, as arising from (PN)MOL-discretized PDEs.

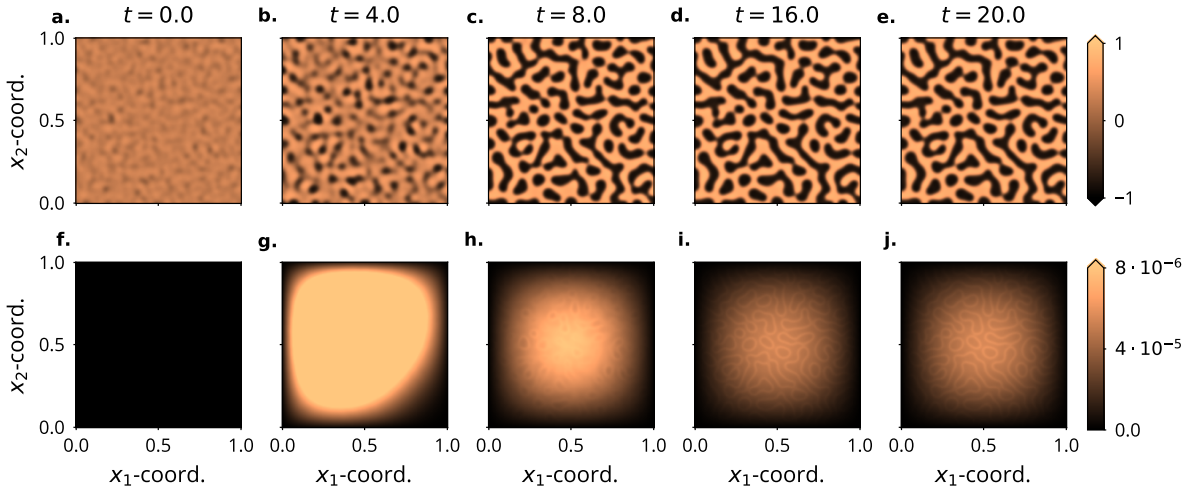


Figure 5.1: **Simulating a high-dimensional ODE:** Probabilistic solution of a discretized FitzHugh-Nagumo PDE model (Ambrosio and Françoise, 2009). Means (a-e) and standard-deviations (f-j), $t_0 = 0$ (left) to $t_{\max} = 20$ (right). The patterns in the uncertainties match those in the solution. The simulated ODE is 125k-dimensional.

This chapter discusses a class of algorithms that computes the solution of initial value problems based on ODEs, of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = f(t, \boldsymbol{x}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0. \tag{5.1}$$

for all $t \in [t_0, t_{\max}]$, $\boldsymbol{x}(t) \in \mathbb{R}^d$. Usually, f is nonlinear, in which case the solution of Equation (5.1) cannot generally be derived in closed form and has to be approximated numerically, as discussed in Section 2.1.3. We continue the work of PN algorithms for ODEs. Like other filtering-based ODE solvers, the algorithm used herein translates numerical approximation of ODE solutions to a problem of probabilistic inference. The resulting (approximate) posterior distribution quantifies the uncertainty associated with the unavoidable discretization error (Bosch et al., 2021) and provides a language that integrates well with other data inference schemes (Kersting et al., 2020a; Schmidt et al., 2021). The main difference to prior work is that we focus on the setting where the dimension d of the ODE is high, that is, say, $d \gg 100$. (It is not clearly defined at which point an ODE counts as high-dimensional, but $d \approx 100$ is already a scale of problems in which previous state-of-the-art probabilistic ODE solvers faced computational challenges.)

Motivation and impact High-dimensional ODEs describe the interaction of large networks of dynamical systems and appear in many disciplines in the natural sciences. The perhaps most prominent example arises in the simulation of discretized partial differential equations (c.f. Chapter 4). There, the dimension of the ODE equals the number of grid points used to discretize the problem (with e.g. finite differences; Schiesser, 2012). More recently, ODEs gained popularity in machine learning through the advent of neural ODEs (Chen et al., 2018), continuous normalising flows (Grathwohl et al., 2018), or physics-informed neural networks (Raissi et al., 2019). With the growing complexity of the model, each of the above can quickly become high-dimensional. If such use cases shall gain from probabilistic solvers, fast algorithms for large ODE systems are crucial.

Prior work and state-of-the-art Many non-probabilistic ODE solvers, for example, explicit Runge–Kutta methods, have a computational complexity linear in the ODE dimension d (Hairer et al., 1993). Explicit Runge–Kutta methods are often the default choices in ODE solver software packages. Compared to the efficiency of the methods provided by DifferentialEquations.jl (Rackauckas and Nie, 2017), SciPy (Virtanen et al., 2020), or Matlab (Shampine and Reichelt, 1997), probabilistic methods have lacked behind so far. Intuitively, ODE filters are a fusion of ODE solvers and Gaussian process models – two classes of algorithms that suffer from high dimensionality. More precisely, the problem is that probabilistic solvers require matrix-matrix operations at each step. The matrices have $O(d^2)$ entries, which leads to $O(d^3)$ complexity for a single solver step and has made the solution of high-dimensional ODEs impossible. ODE filters are essentially nonlinear, approximate Gaussian process inference schemes (with a lot of structure). As in the GP community (e.g. Quiñonero-Candela and Rasmussen, 2005), the path to low computational cost in these models is via factorization assumptions.

Contributions Our main contribution is to prove in which settings ODE filters admit an implementation in $O(d)$ complexity. Thereby, they become a class of algorithms comparable to explicit Runge–Kutta methods not only in estimation performance (error contraction as a function of evaluations of f; Kersting et al., 2020b; Tronarp et al., 2021) but also in computational complexity (cost per evaluation of f). The resulting algorithms deliver uncertainty quantification and other benefits of probabilistic ODE solvers on high-dimensional ODEs (see Figure 5.1. The ODE from this figure will be explored in more detail in Section 5.3). The key novelties of the present chapter are threefold:

1. *Acceleration via independence:* A-priori, ODE filters commonly assume independent ODE dimensions (e.g. Kersting et al., 2020b). We single out those inference schemes that naturally preserve independence. Identification of independence-preserving ODE solvers is helpful because each ODE dimension can be updated separately. The performance implications are that a single matrix-matrix operation with $O(d^2)$ entries is replaced with d matrix-matrix operations with $O(1)$ entries. In other words, $O(d)$ instead of $O(d^3)$ complexity for a single solver step. This is Proposition 15.

2. *Calibration of multivariate output-scales:* A single ODE system often models the interaction between states that occur on different scales. It is useful to acknowledge differing output scales in the "diffusivity" of the prior (details below). We generalise the calibration result by Bosch et al. (2021) to the class of solvers that preserve the independence of the dimensions. This is Proposition 13.

3. *Acceleration via Kronecker structure:* Sometimes, prior independence assumptions may be too restrictive. For instance, one might have prior knowledge of correlations between ODE dimensions (Example 16 in Section 5.2). Fortunately, a subset of probabilistic ODE solvers can exploit *and preserve* Kronecker structure in the system matrices of the state space. Preserving the Kronecker structure brings over the performance gains from above to dependent priors. This is Proposition 17.

Additional minor contributions are detailed where they occur. To demonstrate the scalability of the resulting algorithm, the experiments in Section 5.3 showcase simulations of ODEs with dimension $d \sim 10^7$. Before reading the following sections, it is recommended that the reader is familiar with Section 2.4, which introduces filtering-based probabilistic ODE solvers and notation that is used hereafter.

5.1 Independent Prior Models Accelerate ODE Solvers

This section establishes the main idea of the present chapter: *probabilistic ODE solvers are fast and efficient when the prior models each dimension independently.*

5.1.1 Assumptions

Independent dimensions stem from a diagonal $\mathbf{\Gamma}$.

Assumption 5.1. *Assume that the diffusion* $\mathbf{\Gamma}$ *of the Wiener process in Equation* (2.94) *is a diagonal matrix.*

Assumption 5.1 implies that the initial covariance $\boldsymbol{C}_0$ (Equation (2.95)) is the Kronecker product of a diagonal matrix with another matrix, thus block diagonal. Assumption 5.1 is not very restrictive; in prior work on ODE filters, $\mathbf{\Gamma}$ was always either $\mathbf{\Gamma} = \gamma^2 \boldsymbol{I}$ for some $\gamma > 0$ (Schober et al., 2019a; Tronarp et al., 2019; Kersting et al., 2020b; Bosch et al., 2021; Tronarp et al., 2021; Krämer and Hennig, 2020), or diagonal (Bosch et al., 2021).

5.1.2 Calibration

Tuning the diffusion $\mathbf{\Gamma}$ is crucial to obtain accurate posterior uncertainties. As announced in Section 2.4, the mathematical assumptions for calibrating $\mathbf{\Gamma}$ coincide with the assumptions that lead to an efficient ODE filter. Thus, we discuss $\mathbf{\Gamma}$ before proving the linear complexity of probabilistic solvers under Assumption 5.1.

Four approaches Recall the observed random variable $\mathbf{z}_n$ (Equation (2.109)). ODE filters calibrate $\mathbf{\Gamma}$ with quasi-maximum-likelihood-estimation (quasi-MLE): Consider the prediction error decomposition (Schweppe, 1965),

$$p(\{\mathbf{z}_n\}_{n=0}^N) = p(\mathbf{z}_0) \prod_{n=0}^{N-1} p(\mathbf{z}_{n+1} \mid \mathbf{z}_n) \tag{5.2a}$$

$$\approx \mathcal{N}(\mathbf{z}_0; \boldsymbol{z}_0, \boldsymbol{S}_0) \prod_{n=0}^{N-1} \mathcal{N}(\mathbf{z}_{n+1}; \boldsymbol{z}_{n+1}, \boldsymbol{S}_{n+1}). \tag{5.2b}$$

$\mathbf{\Gamma}$ is a quasi-MLE if it maximises Equation (5.2b). The specific choice of calibration depends on respective model for $\mathbf{\Gamma}$, and reduces to one of four approaches: on the one hand, fixing and calibrating a *time-constant* $\mathbf{\Gamma}$ versus allowing a *time-varying* $\mathbf{\Gamma}$; on the other hand, choosing a *scalar* diffusion $\mathbf{\Gamma} = \gamma^2 \boldsymbol{I}$ versus choosing a *vector-valued* diffusion $\mathbf{\Gamma} = \mathrm{diag}(\gamma^1, ..., \gamma^d)$. Roughly speaking, a time-varying, vector-valued diffusion allows for the greatest flexibility in the probabilistic model. One contribution of the present work is to extend the vector-valued diffusion results by Bosch et al. (2021) to a slightly broader class of solvers (Proposition 13 below).

Time-varying diffusion Allowing $\mathbf{\Gamma}$ to change over the time-steps, all measurements before time t_n are independent of $\mathbf{\Gamma}_n$. Under the assumption of an error-free previous

state (which is common for hyperparameter calibration in ODE solvers), a local quasi-MLE for $\mathbf{\Gamma}_n = \gamma_n^2 \boldsymbol{I}$ arises as (Schober et al., 2019a)

$$\hat{\gamma}_n^2 := \frac{1}{d} \boldsymbol{z}_n^\top \left[\boldsymbol{H}(t_n) \boldsymbol{Q}(h_n) \boldsymbol{H}(t_n)^\top \right]^{-1} \boldsymbol{z}_n. \tag{5.3}$$

This can be extended to a quasi-MLE for the EK0 with vector-valued $\mathbf{\Gamma}_n = \mathrm{diag}(\gamma_n^1, ..., \gamma_n^d)$ (Bosch et al., 2021)

$$(\hat{\gamma}_n^i)^2 := (z_n^i)^2 / [\boldsymbol{H}(t_n) \boldsymbol{Q}(h_n) \boldsymbol{H}(t_n)^\top]_{ii}, \tag{5.4}$$

for all $i = 1, ..., d$. In this work, we generalise the EK0 requirement to Assumption 5.1 and a diagonal Jacobian.

Proposition 13. *Under Assumption 5.1 and for diagonal* $\mathrm{D}f(\boldsymbol{x})$*, the estimators* $(\hat{\gamma}_n^i)_i$ *in Equation* (5.4) *are quasi-MLEs.*

Sketch of the proof. Two ideas are relevant: (i) a diagonal Jacobian implies a block diagonal $\boldsymbol{H}(t_n)$ and a diagonal $\boldsymbol{H}(t_n)\boldsymbol{Q}(h_n)\boldsymbol{H}(t_n)^\top$ (which will be proved formally in Proposition 15 below); (ii) the local evidence, i.e. the probability of $\mathcal{N}(\boldsymbol{z}_n, \boldsymbol{S}_n)$ being zero, decomposes into a sum over the coordinates. Maximising each summand with respect to γ_n^i yields the claim. □

Proposition 13 is a generalisation of the results by Bosch et al. (2021) in the sense that Proposition 13 is not restricted to the EK0. A very similar case can be made for time-constant diffusion.

Proposition 14. *Under Assumption 5.1 and for diagonal* $\mathrm{D}f(\mathbf{x})$*, a quasi-maximum likelihood estimate for a vector-valued, time-constant diffusion model* $\mathbf{\Gamma} = \mathrm{diag}((\gamma^1)^2, ..., (\gamma^d)^2)$ *is given by the estimator*

$$(\hat{\gamma}^i)^2 := \frac{1}{N} \sum_{i=1}^{N} \frac{(z_n^i)^2}{[\boldsymbol{S}_n]_{ii}}, \qquad i = 1, \ldots, d, \tag{5.5}$$

where $\boldsymbol{S}_n := \boldsymbol{H}(t_n)\boldsymbol{Q}(h_n)\boldsymbol{H}(t_n)^\top$ *is the diagonal covariance matrix of the measurement* $\mathbf{z}_n$ *(recall Section 2.4.2).*

Proof. The proof is structured as follows. First, we show that an initial covariance

$$\boldsymbol{C}_0 = \mathrm{blockdiag}((\gamma^1)^2 \breve{\boldsymbol{C}}_0, \ldots, (\gamma^d)^2 \breve{\boldsymbol{C}}_0) \tag{5.6}$$

implies covariances

$$\boldsymbol{C}_n^- = \text{blockdiag}\left((\gamma^1)^2(\boldsymbol{C}_n^1)^-, \ldots, (\gamma^d)^2(\boldsymbol{C}_n^d)^-\right), \tag{5.7a}$$
$$\boldsymbol{C}_n = \text{blockdiag}\left((\gamma^1)^2\boldsymbol{C}_n^1, \ldots, (\gamma^d)^2\boldsymbol{C}_n^d\right), \tag{5.7b}$$
$$\boldsymbol{S}_n = \text{diag}\left((\gamma^1)^2 s_n^1, \ldots, (\gamma^d)^2 s_n^d\right). \tag{5.7c}$$

Then, for measurement covariances $\boldsymbol{S}_n$ of such form, we can compute the (quasi) maximum likelihood estimate $\hat{\boldsymbol{\Gamma}}$. Because every covariance depends multiplicatively on γ, calibration can happen entirely post-hoc.

Block-wise scalar diffusion Recall from Section 2.4.1 that the transition matrix and the process noise covariance are of the form $\boldsymbol{\Phi}(h_n) = \boldsymbol{I}_d \otimes \breve{\boldsymbol{\Phi}}(h_n)$ and $\boldsymbol{Q}(h_n) = \boldsymbol{\Gamma} \otimes \breve{\boldsymbol{Q}}(h_n)$. Thus, for a diagonal diffusion $\boldsymbol{\Gamma} = \text{diag}((\gamma^1)^2, ..., (\gamma^d)^2)$, both $\boldsymbol{\Phi}(h_n)$ and $\boldsymbol{Q}(h_n)$ are block diagonal. Assuming a block diagonal covariance matrix that depends multiplicatively on γ,

$$\boldsymbol{C}_{n-1} = \text{blockdiag}\left((\gamma^1)^2\boldsymbol{C}_{n-1}^1, \ldots, (\gamma^d)^2\boldsymbol{C}_{n-1}^d\right), \tag{5.8}$$

the extrapolated covariance is also of the form

$$\boldsymbol{C}_n^- = \text{blockdiag}\left((\gamma^1)^2(\boldsymbol{C}_n^1)^-, \ldots, (\gamma^d)^2(\boldsymbol{C}_n^d)^-\right), \tag{5.9a}$$
$$(\boldsymbol{C}_n^i)^- := \breve{\boldsymbol{\Phi}}(h_n)\boldsymbol{C}_{n-1}^i\breve{\boldsymbol{\Phi}}(h_n)^\top + \breve{\boldsymbol{Q}}(h_n), \qquad i = 1, \ldots, d. \tag{5.9b}$$

The diagonal Jacobian $\mathrm{D}f(\mathbf{x})$ implies a block diagonal linearization matrix

$$\boldsymbol{H}_n = \boldsymbol{E}_1 - \mathrm{D}f(\mathbf{x})\boldsymbol{E}_0 = \text{blockdiag}\left(\boldsymbol{H}_n^1, \ldots, \boldsymbol{H}_n^d\right), \tag{5.10a}$$
$$\boldsymbol{H}_n^i := \boldsymbol{e}_1 - [\mathrm{D}f(\mathbf{x})]_{i,i}\boldsymbol{e}_0, \qquad i = 1, \ldots, d. \tag{5.10b}$$

The measurement covariance $\boldsymbol{S}_n$ is therefore given by a *diagonal* matrix and depends multiplicatively on γ, as

$$\boldsymbol{S}_n = \boldsymbol{H}_n\boldsymbol{C}_n^-\boldsymbol{H}_n^\top = \text{diag}\left((\gamma^1)^2 s_n^1, \ldots, (\gamma^d)^2 s_n^d\right), \tag{5.11a}$$
$$s_n^i := \boldsymbol{H}_n^i(\boldsymbol{C}_n^i)^-(\boldsymbol{H}_n^i)^\top, \qquad i = 1, \ldots, d. \tag{5.11b}$$

This implies a block diagonal Kalman gain

$$\boldsymbol{\Xi}_n = \boldsymbol{I} - \boldsymbol{C}_n^-\boldsymbol{H}(t_n)^\top\boldsymbol{S}_n^{-1}\boldsymbol{H}(t_n) = \text{blockdiag}\left(\boldsymbol{\Xi}_n^1, \ldots \boldsymbol{\Xi}_n^d\right), \tag{5.12a}$$
$$\boldsymbol{\Xi}_n^i := \boldsymbol{I}_{\nu+1} - (\boldsymbol{C}_n^-)^i(\boldsymbol{H}_n^i)^\top\boldsymbol{H}_n^i/s_n^i \qquad i = 1, \ldots, d. \tag{5.12b}$$

Finally, we obtain the corrected covariance

$$\boldsymbol{C}_n = \text{blockdiag}\left((\gamma^1)^2\boldsymbol{C}_n^1, \ldots, (\gamma^d)^2\boldsymbol{C}_n^d\right), \tag{5.13a}$$

$$\boldsymbol{C}_n^i := \boldsymbol{\Xi}_n^i(\boldsymbol{C}_n^i)^-(\boldsymbol{\Xi}_n^i)^\top \qquad i = 1, \ldots, d. \tag{5.13b}$$

This concludes the first part of the proof.

Computing the quasi-MLE Now it is still left to compute the (quasi-) MLE $\hat{\boldsymbol{\Gamma}} = \text{diag}((\hat{\gamma}^1)^2, ..., (\hat{\gamma}^d)^2)$ by maximizing the log-likelihood

$$\log p(\boldsymbol{z}_{1:N}) = \log \prod_{n=1}^{N} \mathcal{N}\left(\mathbf{0}; \boldsymbol{z}_n, \boldsymbol{S}_n\right). \tag{5.14}$$

Since $\boldsymbol{S}_n = \text{diag}\left((\gamma^1)^2 s_n^1, \ldots, (\gamma^d)^2 s_n^d\right)$ is a diagonal matrix, we obtain

$$\hat{\boldsymbol{\Gamma}} = \arg\max_{\boldsymbol{\Gamma}} \sum_{n=1}^{N} \log \mathcal{N}\left(\mathbf{0}; \boldsymbol{z}_n, \boldsymbol{S}_n\right) \tag{5.15a}$$

$$= \arg\max_{\boldsymbol{\Gamma}} \sum_{i=1}^{d} \left(-\frac{N \log(\hat{\gamma}^i)^2}{2} - \sum_{n=1}^{N} \frac{(\boldsymbol{z}_n)_d^2}{2 s_n^i (\hat{\gamma}^i)^2} \right). \tag{5.15b}$$

By taking the derivative and setting it to zero, we obtain the quasi-MLE from Equation (5.5).

□

5.1.3 Complexity

Now, with calibration in place, we can discuss the computational complexity of ODE filters under Assumption 5.1. The following proposition establishes that for diagonal Jacobians, a single solver step costs $O(d)$.

Proposition 15. *Suppose that Assumption 5.1 is in place. If the Jacobian of the ODE is (approximated as) a diagonal matrix, then a single step with a filtering-based probabilistic ODE solver costs $O(d\nu^3)$ in floating-point operations, and $O(d\nu^2)$ in memory.*

Proof. Let $\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{m}_n, \boldsymbol{C}_n)$. Assume that $\boldsymbol{C}_n$ is block diagonal. We show that block diagonality is preserved through a step, and since by Assumption 5.1, $\boldsymbol{C}_0$ is block diagonal, we do not lose generality. Recall $\boldsymbol{\Phi}(h_n)$ and $\boldsymbol{Q}(h_n)$ from Equations (2.97) and (2.98). $\boldsymbol{\Phi}(h_n)$ is block diagonal, and since $\boldsymbol{\Gamma}_n$ is diagonal, $\boldsymbol{Q}(h_n)$ is block diagonal.

(i) Extrapolate the mean: The mean is extrapolated according to Equation (2.102a), which costs $O(d\nu^2)$, because of the block diagonal $\boldsymbol{\Phi}(h_n)$. Each dimension is extrapolated independently.

(ii) Evaluate the ODE: Next, $\boldsymbol{H} = \boldsymbol{H}(t_{n+1})$ and $\boldsymbol{b} = \boldsymbol{b}(t_{n+1})$ from Equation (2.107) are assembled, which involves evaluating f and $\mathrm{D}f(\boldsymbol{x})$ at $\boldsymbol{\xi} := \boldsymbol{E}_0 \boldsymbol{m}^-_{n+1}$ ($\boldsymbol{E}_0$ is a projection matrix and can be implemented as array indexing, so $\boldsymbol{\xi}$ comes at negligibily low cost). $\mathrm{D}f(\boldsymbol{x}) = \mathrm{diag}([\mathrm{D}f(\boldsymbol{x})]^1, ..., [\mathrm{D}f(\boldsymbol{x})]^d)$ is a diagonal matrix, therefore

$$\boldsymbol{H} = \mathrm{blockdiag}(\boldsymbol{H}^1, ..., \boldsymbol{H}^d) \tag{5.16}$$

is block diagonal with blocks

$$\boldsymbol{H}^i := \boldsymbol{e}_1 - \boldsymbol{e}_0 \left[\mathrm{D}f(\boldsymbol{x})\right]^i, \quad i = 1, ..., d \tag{5.17}$$

(recall the basis vectors $\boldsymbol{e}_q$ from Equation (2.96)). The block diagonal $\boldsymbol{H}$ has been pre-empted in Proposition 13 above.

(iii) Calibrate $\boldsymbol{\Gamma}$: The cost of assembling the quasi-MLE for $\boldsymbol{\Gamma}_{n+1}$ according to Equation (5.3) or Equation (5.4) is $O(d)$, because the matrix to be inverted is diagonal.

(iv) Extrapolate the covariance: The covariance can be extrapolated dimension-by-dimension as well, because $\boldsymbol{C}_n$, $\boldsymbol{\Phi}(h_n)$, and $\boldsymbol{Q}(h_n)$ are all block diagonal with the same block structure: d square blocks with $\nu + 1$ rows and columns; recall Equation (2.102b). In reality, the matrix-matrix multiplication is replaced by a QR decomposition; we refer to Section 2.4.3 for details on square-root implementation. Using either strategy – square-root or traditional implementation – extrapolating the covariance costs $O(d\nu^3)$ and $\boldsymbol{C}^-_{n+1}$ is block diagonal.

(v) Measure: Computing the mean of $\mathbf{z}_{n+1}$ (recall Equations (2.109) and (2.110)) costs $O(d)$. The covariance $\boldsymbol{S}_{n+1}$ of $\mathbf{z}_{n+1}$ is diagonal, since $\boldsymbol{H}$ and $\boldsymbol{C}^-_{n+1}$ are block diagonal. Thus, assembling *and inverting* $\boldsymbol{S}_{n+1}$ costs $O(d)$.

(vi) Correct mean and covariance: The mean is corrected according to Equation (2.111b), which – since $\boldsymbol{S}_{n+1}$ is diagonal – costs $O(d\nu)$. The covariance is corrected according to Equations (2.111c) and (2.111d), the complexity of which hinges on the structure of $\boldsymbol{\Xi}$ (Equation (2.111d)): due to the block diagonal $\boldsymbol{C}^-_{n+1}$, $\boldsymbol{H}$, and $\boldsymbol{S}_{n+1}$, $\boldsymbol{\Xi}$ is block diagonal again, and correcting the covariance costs $O(d\nu^3)$. The square-root matrix of $\boldsymbol{C}_{n+1}$ arises by multiplying $\boldsymbol{\Xi}$ with the "left" square-root matrix of $\boldsymbol{C}^-_{n+1}$. The complexity remains the same (asymptotically, though QR decompositions cost more than matrix multiplications).

All in all, ODE filter steps preserve block-diagonal structure in the covariances. The expensive phases are the covariance extrapolation and correction in $O(d\nu^3)$ floating-point operations. The maximum memory demand is $O(d\nu^2)$ for the block diagonal covariances. □

While it may seem restrictive at first to use only the diagonal of the Jacobian, Proposition 15 includes the EK0, one of the central ODE filters. The $O(d)$ complexity puts the EK0 and the diagonal EK1 into the complexity class of explicit Runge–Kutta methods. Usually, $\nu < 12$ holds (Krämer and Hennig, 2020).

5.2 EK0 Preserves Kronecker Structure

Scalar or diagonal diffusion may be too restrictive in certain situations.

Example 16. *Consider a spatio-temporal Gaussian process model* $\mathbf{u}(t, x) \sim \mathcal{GP}(0, \gamma^2 k_t \otimes k_x)$*, where* k_t *is the covariance kernel that directly corresponds to an integrated Wiener process prior (Särkkä and Solin, 2019). Such a spatiotemporal model could be a useful prior distribution for applying an ODE solver to problems that are discretized PDEs, because* k_x *encodes spatial dependency structures. Restricted to a spatial grid* $\mathbb{X} := \{x_1, ..., x_G\}$*,* $\mathbf{u}(t, \mathbb{X})$ *satisfies the prior model in Equations* (2.94) *and* (2.95)[1]*, but with* $\mathbf{\Gamma} = \gamma^2 k_x(\mathbb{X}, \mathbb{X})$ *(Solin, 2016), which is usually dense.*

5.2.1 Assumptions

Despite the lack of independence in Example 16, fast ODE solutions remain possible with the EK0. In the remainder of this section, let $\mathbf{\Gamma} = \gamma^2 \breve{\mathbf{\Gamma}}$ for some matrix $\breve{\mathbf{\Gamma}}$ and some scalar γ. Calibrating the scalar γ allows preserving Kronecker structure in the system matrices that appear in an ODE filter step (Proposition 17 below). Tronarp et al. (2019) show how for $\mathbf{\Gamma} = \gamma^2 \breve{\mathbf{\Gamma}}$, a time-constant quasi-MLE $\hat{\gamma}$ arises in closed form and also, that the posterior covariances all look like $\boldsymbol{C}_n = \gamma^2 \breve{\boldsymbol{C}}_n$: calibration can happen entirely post-hoc.

Constraints The following statement about linear complexity of ODE filters is only valid under two constraints: one can ignore (i) the quadratic costs of multiplying the posterior covariances with the quasi-MLE, and (ii) the cubic costs of solving a linear system involving $\mathbf{\Gamma}$. The matrices $\mathbf{\Phi}, \boldsymbol{Q}, \boldsymbol{C}_0$ are all Kronecker products of a $\mathbb{R}^{d \times d}$ ("left") and a $\mathbb{R}^{\nu \times \nu}$ factor ("right"). The first constraint is thus avoided by scaling the "right" Kronecker factor of the covariances with γ^2 in $O(\nu^2)$. (ODE filters preserve Kronecker structure; see below.) The second one becomes the following assumption.

Assumption 5.2. *Assume that the inverse of* $\mathbf{\Gamma}$ *is readily available and cheap to apply; that is, the quantity* $\boldsymbol{x}^\top \mathbf{\Gamma}^{-1} \boldsymbol{x}$ *can be computed in* $O(d)$*.*

Naturally, Assumption 5.2 holds for diagonal or at least sufficiently sparse matrices $\mathbf{\Gamma}$. There are also settings in which Assumption 5.2 holds even if $\mathbf{\Gamma}$ is dense. For instance, if $\mathbf{\Gamma}$ is the covariance of a Gauss–Markov random field, the sparsity structure in $\mathbf{\Gamma}^{-1}$

[1] Technically, the stack of $\mathbf{x}$ and its ν derivatives does.

implies adjacency of grid nodes (Lindgren et al., 2011; Sidén and Lindsten, 2020). In Example 16 with a spatial Matern kernel, for example, inverse Gram matrices can be approximated efficiently using the stochastic partial differential equation formulation (Lindgren et al., 2011).

5.2.2 Computational Complexity

Under Assumption 5.2, a single EK0 step costs $O(d)$:

Proposition 17. *Under Assumption 5.2, and if a time-constant diffusion model $\mathbf{\Gamma} = \gamma^2 \breve{\mathbf{\Gamma}}$ is calibrated via γ, a single step of the EK0 costs $O(\nu^3 + d\nu^2)$ floating point operations, and $O(d\nu + d^2 + \nu^2)$ memory.*

Proof. The proof parallels that of Proposition 15. It hinges on computing everything only in the "right" factor of each Kronecker matrix.

Let $\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{m}_n, \boldsymbol{C}_n)$. Assume $\boldsymbol{C}_n = \mathbf{\Gamma} \otimes \breve{\boldsymbol{C}}_n$ which is no loss of generality, because such a Kronecker structure is preserved through the ODE filter step as shown below.

(i) Extrapolate mean: The mean extrapolation costs $O(d\nu^2)$ like in the proof of Proposition 15.

(ii) Evaluate the ODE: Evaluation of $\boldsymbol{H}$ and $\boldsymbol{b}$ is essentially free – recall that we only consider the EK0 in this setting, which uses the projection $\boldsymbol{H}(t_n) = \boldsymbol{E}_1$. Matrix multiplication with $\boldsymbol{H}$ consists of a projection, which costs $O(1)$.

(iii) Calibrate: Calibration of a time-constant γ^2 costs $O(d)$ under Assumption 5.2.

(iv) Extrapolate covariance: In the time-constant diffusion model, $\boldsymbol{Q}(h_n)$ and $\boldsymbol{C}_n$ are both Kronecker matrices and share the left Kronecker factor: $\mathbf{\Gamma}$. Thus, the extrapolation of the covariance can be carried out "in the right Kronecker factor", which costs $O(\nu^3)$ in traditional as well as square-root implementation. Denote the extrapolated covariance by $\boldsymbol{C}^-_{n+1} := \mathbf{\Gamma} \otimes \breve{\boldsymbol{C}}^-_{n+1}$.

(v) Measure: Recall $\boldsymbol{H}(t_n) = \boldsymbol{E}_1 = \boldsymbol{I} \otimes \boldsymbol{e}_1$. The mean of the measured random variable $\mathbf{z}_n \sim \mathcal{N}(\boldsymbol{z}_n, \boldsymbol{S}_n)$ comes at negligible cost. The covariance

$$\boldsymbol{S}_{n+1} = \boldsymbol{H}(t_{n+1}) \boldsymbol{C}^-_{n+1} \boldsymbol{H}(t_{n+1})^\top = \mathbf{\Gamma} \otimes \left[\boldsymbol{e}_1 \breve{\boldsymbol{C}}^-_{n+1} \boldsymbol{e}_1^\top \right] \tag{5.18}$$

requires a single element in $\breve{\boldsymbol{C}}^-_{n+1}$. The Kalman gain

$$\boldsymbol{K} := \boldsymbol{C}^-_{n+1} \boldsymbol{H}(t_{n+1})^\top \boldsymbol{S}_{n+1}^\top = \boldsymbol{I} \otimes \breve{\boldsymbol{K}}, \tag{5.19}$$

with $\breve{\boldsymbol{K}} := \boldsymbol{e}_1 \breve{\boldsymbol{C}}^-_{n+1} / \left[\boldsymbol{e}_1 \breve{\boldsymbol{C}}^-_{n+1} \boldsymbol{e}_1^\top \right]$ involves dividing the first row of $\breve{\boldsymbol{C}}^-_{n+1}$ by a scalar. Its cost is $O(\nu + 1)$.

(vi) Correct mean and covariance: The mean is corrected in $O(d\nu^2)$ as in the proof of Proposition 15. Due to the Kronecker structure in $\boldsymbol{K}$, the "left" Kronecker factor of $\boldsymbol{C}_{n+1}$ must be $\boldsymbol{\Gamma}$ again. Therefore, we need to correct only the "right" Kronecker factor in $O(\nu^3)$.

All in all, under the assumption of cheap calibration, a single step with the EK0 costs $O(d\nu^2)$ and the expensive steps are (as before) the covariance extrapolation and the covariance correction. The total memory costs are the requirements of storing $\boldsymbol{\Gamma}$, the mean(s) in $O(\nu d)$, and the "right" Kronecker factor(s) in $O(\nu^2)$. □

Proposition 17 can be extended to time-varying diffusion if one tracks γ in the "right" Kronecker factor instead of the "left" one. Since this obfuscates the notation, we refer the reader to Krämer et al. (2021a, Supplement D). The quadratic $O(d^2)$ memory requirement is entirely due to the cost of storing $\boldsymbol{\Gamma}$ – if $\boldsymbol{\Gamma}$ or its inverse are banded matrices, for instance, it reduces to $O(d\nu + \nu^2)$.

5.3 Empirical Evaluations

5.3.1 ODE Problems

Experiments will be conducted on different ODE problems with different properties. We begin by introducing each of them below.

Lorenz96 The Lorenz96 model describes a chaotic dynamical system for which the dimension can be chosen freely (Lorenz, 1996). It is given by a system of $N \geq 4$ ODEs

$$\dot{x}_1 = (x_2 - x_{N-1})x_N - x_1 + F, \tag{5.20a}$$
$$\dot{x}_2 = (x_3 - x_N)x_1 - x_2 + F, \tag{5.20b}$$
$$\vdots \tag{5.20c}$$
$$\dot{x}_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F \qquad i = 3, \ldots, N-1, \tag{5.20d}$$
$$\vdots \tag{5.20e}$$
$$\dot{x}_N = (x_1 - x_{N-2})x_{N-1} - x_N + F, \tag{5.20f}$$

with forcing term $F = 8$, initial values $x_1(0) = F + 0.01$ and $x_{>1}(0) = F$, and time span $t \in [0, 30]$.

Pleiades The Pleiades system describes the motion of seven stars in a plane, with coordinates (x_i, y_i) and masses $m_i = i$, $i = 1, \ldots, 7$ (Hairer et al., 1993, Section II.10).

It can be described with a system of 28 ODEs

$$\dot{x}_i = v_i \tag{5.21a}$$
$$\dot{y}_i = w_i \tag{5.21b}$$
$$\dot{v}_i = \sum_{j \neq i} m_j (x_j - x_i)/r_{ij}, \tag{5.21c}$$
$$\dot{w}_i = \sum_{j \neq i} m_j (y_j - y_i)/r_{ij}, \tag{5.21d}$$

where $r_{ij} = ((x_i - x_j)^2 + (y_i - y_j)^2)^{3/2}$, for $i, j = 1, \ldots, 7$. It is commonly solved on the time span $t \in [0, 3]$ and with initial locations

$$x(0) = [3, 3, -1, -3, 2, -2, 2], \tag{5.22a}$$
$$y(0) = [3, -3, 2, 0, 0, -4, 4], \tag{5.22b}$$
$$v(0) = [0, 0, 0, 0, 0, 1.75, -1.5], \tag{5.22c}$$
$$w(0) = [0, 0, 0, -1.25, 1, 0, 0]. \tag{5.22d}$$

FitzHugh–Nagumo PDE Let $\Delta = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}$ be the Laplacian. The FitzHugh–Nagumo partial differential equation (PDE) is (Ambrosio and Françoise, 2009)

$$\frac{\partial}{\partial t} u(t, x) = a \Delta u(t, x) + u(t, x) - u(t, x)^3 - v(t, x) + k, \tag{5.23a}$$
$$\frac{\partial}{\partial t} v(t, x) = \frac{1}{\tau} (b \Delta v(t, x) + u(t, x) - v(t, x)) \tag{5.23b}$$

for $x \in [0, 1] \times [0, 1] \subseteq \mathbb{R}^2$, some parameters a, b, k, τ, and initial values $u(t_0, x) = h_0(x)$, $v(t_0, x) = h_1(x)$. In our experiments, we chose $a = 208 \cdot 10^{-4}$, $b = 5 \cdot 10^{-3}$, $k = -5 \cdot 10^{-3}$, $\tau = 0.1$. As initial values, we used random samples from the uniform distribution on $(0, 1)$. We solve it from $t_0 = 0$ to $t_{\max} = 20$. To turn the PDE into a system of ODEs, we discretized the Laplacian with central, second-order finite differences schemes on a uniform grid. The mesh size of the grid determines the number of grid points, which controls the dimension of the ODE problem.

Van der Pol The Van der Pol system is often employed to evaluate the stability of stiff ODE solvers (Wanner and Hairer, 1996). It is given by a system of ODEs

$$\dot{x}_1(t) = x_2(t), \qquad \dot{x}_2(t) = \mu \left(\left(1 - x_1^2(t)\right) x_2(t) - x_1(t) \right), \tag{5.24}$$

with stiffness constant $\mu > 0$, time span $t \in [0, 6.3]$, and initial value $x(0) = [2, 0]$.

5.3.2 Experiments

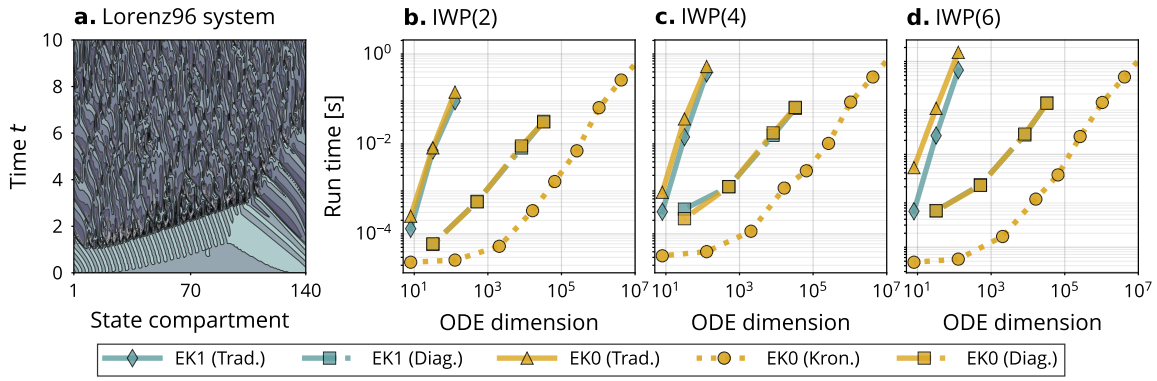
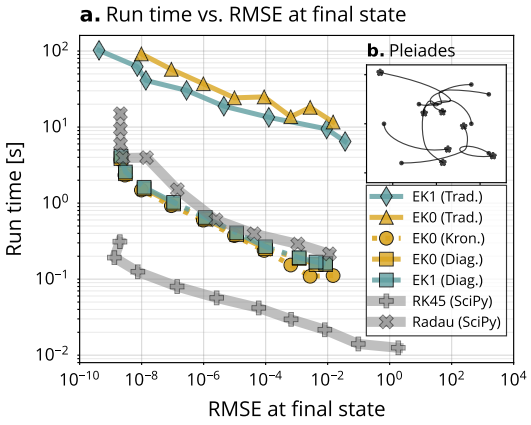


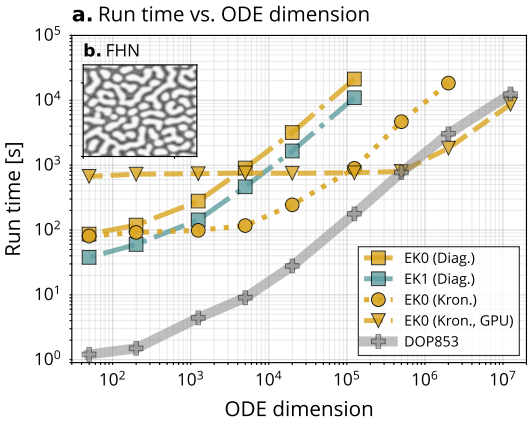
Figure 5.2: **Runtime of a single ODE filter step:** Run time (wall-clock) of a single step of ODE filter variations on the Lorenz96 problem (a) for increasing ODE dimension and $\nu = 2, 4, 6$ (b-d). The traditional implementations cost $O(d^3)$ per step, the diagonal EK1 and diagonal EK0 are $O(d)$ per step, just like the Kronecker version of the EK0. The Kronecker EK0 is significantly faster than the diagonal version(s).

A single ODE filter step We begin by evaluating the cost of a single step of the ODE filter variations on the Lorenz96 problem. This is a chaotic dynamical system and recommends itself for the first experiment, as its dimension can be increased freely. We time a single ODE filter step for increasing ODE dimension d and different solver orders $\nu \in \{2, 4, 6\}$. The results are depicted in Figure 5.2. The traditional EK0 and EK1 become infeasible due to their cubic complexity in the dimension. The diagonal EK1 and the diagonal EK0 exhibit their $O(d)$ cost. The Kronecker EK0 is cheaper than the independence-based solvers. A step with the Kronecker EK0 takes $\sim$1 second for a 16 million-dimensional ODE on a generic, consumer-level CPU. Altogether, Figure 5.2 confirms Propositions 15 and 17.

A full simulation Next, we evaluate whether the performance gains for a single ODE filter step translate into a reduced overall runtime (including step-size adaptation and calibration) on a medium-dimensional problem: the Pleiades problem (Hairer et al., 1993). It describes the motion of seven stars in a plane and is commonly solved as a system of 28 first-order ODEs. The results are in Figure 5.3a. Pleiades reveals the increased efficiency of the ODE filters. The probabilistic solvers are as fast as Radau, only by a factor $\sim$10 slower than SciPy's RK45 (Virtanen et al., 2020), but 100 times faster than their reference implementations. (It should be noted that the ODE filters use just-in-time compilation for some components, whereas SciPy does not.)



(a) **Runtime efficiency of fast ODE filters:** Run time per root mean-square error of the ODE filters (a) on the Pleiades problem (b). The figure also shows two reference ODE filters, EK0 and EK1 in the traditional implementation, and Scipy's RK45 (explicit) and Radau (implicit). On the 28-dimensional Pleiades problem, the improved implementation accelerates the ODE filter implementations significantly.



(b) **High-dimensional PDE discretization:** Run-time of ODE filters on the discretized FitzHugh-Nagumo model for increasing ODE dimension (i.e. increasing spatial resolution) including calibration and adaptive time-steps. SciPy's DOP853 for reference. Simulating $\gg 10^6$-dimensional ODEs takes $\approx 3h$.

Figure 5.3: **Evaluating the runtime of fast ODE filters.**

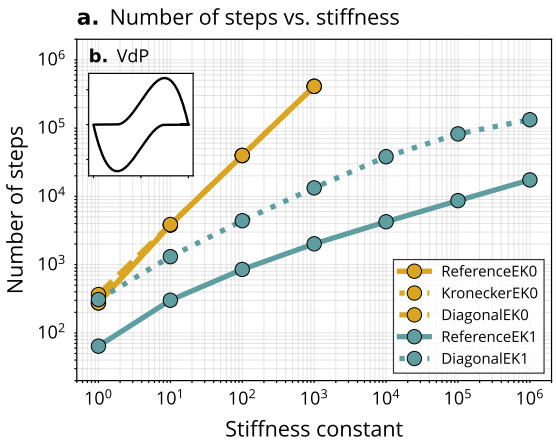


Figure 5.4: **Evaluating the stability of fast ODE filters.** Number of steps taken by an ODE filter (a) for an increasingly stiff Van der Pol system (b). The diagonal EK1 is more stable than the EK0, but less stable than the EK1 (which is expected because it uses strictly less Jacobian information).

A high-dimensional setting To evaluate how well the improved efficiency translates to extremely high dimensions, we solve the discretized FitzHugh-Nagumo PDE model on high spatial resolution (which translates to high dimensional ODEs). The results are in Figure 5.3b. The main takeaway is that ODEs with millions of dimensions can be solved *probabilistically* within a realistic time frame (hours), which has not been possible before. GPUs improve the runtime for extremely high-dimensional problems ($d \gg 10^5$).

Stability of the diagonal EK1 How much do we lose by ignoring off-diagonal elements in the Jacobian? To evaluate the loss (or preservation) of stability against the A-stable EK1 (Tronarp et al., 2019), we solve the Van der Pol system (Guckenheimer, 1980). It includes a free parameter $\mu > 0$, whose magnitude governs the stiffness of the problem: the larger μ, the stiffer the problem, and for e.g. $\mu = 10^6$, Van der Pol is a famously stiff equation. The results are in Figure 5.4. We observe how the diagonal EK1 is less stable than the reference EK1 for increasing stiffness constant, but also that it is significantly more stable than the EK0, for instance. It is a success that the diagonal EK1 solves the van der Pol equation for large μ.

6 Discussion and Conclusion

This work presented advances in probabilistic numerical methods for simulation of dynamical systems that describe temporal and spatial interactions of its components. Thereby, we (i) investigated how different sources of information can be incorporated naturally into a system, (ii) how spatial diffusion can be added while keeping track of temporal *and* spatial discretization error using similar mechanisms and (iii) how to compute solutions in this framework efficiently, such that these methods can be applied in practice, keeping the computational expenses feasible for practitioners.

By coupling mechanistic and data-driven inference, the algorithm presented in Chapter 3 builds on the core premise of probabilistic numerics – that computation itself is a data source that does not differ, formally, from observational data. Information from observations and mechanistic knowledge (in the form of an ODE) can thus be described in the same language of Bayesian filtering and smoothing. This removes the need for an outer loop over multiple forward solves and thus drastically reduces the computational cost. Our experimental evaluation corroborates that the resulting approximate posterior is close to the ground truth and drastically reduces computational cost over Monte Carlo alternatives. It faithfully captures multiple sources of uncertainty from the data, numerical (discretization) error, and epistemic uncertainty about the mechanism. In particular, concerning the presented experiment on real COVID-19 data, a natural next line of thought goes towards spatial diffusions, which was not considered in this chapter. To take a step towards probabilistic numerical simulations of spatio-temporal dynamics models, Chapter 4 presented probabilistic strategies for discretizing time-dependent PDEs, and for making use of the resulting quantification of spatial discretization uncertainty in a probabilistic ODE solver. We discussed practical considerations, including sparsification of the differentiation matrices, keeping the dimensionality of the state space low, and hyperparameter choices. Altogether, and unlike traditional PDE solvers, the probabilistic method of lines unlocks quantification of spatio-temporal correlations in an approximate PDE solution, all while preserving the efficiency of adaptive ODE solvers. This makes it a valuable algorithm in the toolboxes of probabilistic programs and differential equation solvers and may serve as a backbone for latent force models, inverse problems, and differential-equation-centric machine learning. When working with discretized PDEs it quickly became apparent that probabilistic ODE solvers have to lose their cubic computational complexity with regard to the dimensionality of the state-space model. Instead, for probabilistic ODE solvers to capitalize on their theoretical advantages, their computational cost has to come close to that of their non-probabilistic point-estimate counterparts (which benefit from decades of optimization). High-dimensional problems

are one obstacle on this path, which we tackled in Chapter 5. We showed that independence assumptions in the underlying state-space model, or preservation of Kronecker structures, can bring the computational complexity of a large subset of known ODE filters close to non-probabilistic, explicit Runge–Kutta methods. As a result, probabilistic simulation of extremely large systems of ODEs is now possible, opening up opportunities to exploit the advantages of probabilistic ODE solvers on challenging real-world problems. This entire work revolved around two concepts that are both cornerstones of modern machine learning: numerical computation and statistical inference. Notably, we have never drawn a clear boundary between these concepts. Rather, by treating mechanistic and empirical knowledge with the same tools, this work continued to blur the line between forward and inverse problems. Simulation of complex dynamical systems and discovering latent forces therein amount to surprisingly similar tasks, that of statistically inferring unknown quantities of interest. First steps have been made to tackle large-scale, complex spatio-temporal models while upholding this mindset.

Outlook

The incredibly versatile and elegant language that probabilistic state estimation in state-space models provides, first and foremost, suggests a plethora of applications. This thesis aims at providing some groundwork on the way to establishing PN methods in the sciences and for making them accessible to practitioners – out of the conviction that they provide a tangible benefit. However, in this regard, there is still a long way to go. The range of tremendously relevant questions is vast: climate, geophysics, fluid dynamics, politics, economics, to name only a few areas of research – all of these rely on mathematical descriptions of complex dynamics in order to make predictions. Identifying latent forces acting on these systems is of special interest in real-world applications, especially when aiming to draw conclusions from and make decisions based on predictions of the model at hand. The overarching goal is to create and provide a concise language in which domain knowledge can be captured and integrated meaningfully into a model. Probabilistic state-space models and Gaussian filtering are a good step in this direction, due to their computational and conceptual ease. Overcoming the need for an outer loop around the numerical simulation of large-scale systems makes an agile way of working with these models feasible. The price to be paid is that of making approximations like linearity or Gaussianity. Studying different approximate inference techniques based on the presented state-space models (especially the particle filter (Särkkä, 2013; Naesseth et al., 2019) comes to mind) might yield fruitful results. Implementing these methods robustly and computationally feasible on large-scale models bears challenges on its own.

Another interesting direction might be the combination of parametric and nonparametric methods, in which the state-space model becomes but one part of a larger model. Optimization or sampling routines around a numerical simulation could be built around a probabilistic solver instead, thus sampling or optimizing (lower-dimensional) quantities in the model while the probabilistic ODE solution and the simultaneous latent-force

inference are refined along the way.

In general, the probabilistic spatial discretization for PDEs in Chapter 4 leaves a lot of interesting directions open. Firstly, this method involves many hyperparameters, which significantly impact the outcome and are at the same time difficult to interpret and hence to choose. In particular, the respective covariance functions of the prior spatio-temporal GP could be investigated further and different models could be applied to different scenarios to assess the potential of more elaborate priors. One could proceed to think about a method that uses spatial uncertainty to adaptively refine the spatial discretization grid wherever the uncertainty suggests a resolution that is too coarse. What exactly such adaptive mesh-refinement could look like in this setting is left to the reader's imagination here, but further investigations are called for.

To conclude, we hope that this work and the presented methods pave the way towards a more prominent use of probabilistic numerical algorithms in practical applications. State estimation from data using probabilistic state-space models and approximate inference is not a new idea and has been around for decades. However, we believe that recent advances in PN began to uncover tremendous potential by unifying numerical computation and statistical inference under one language. We hope that this sparks the interest of the reader in using these methods or even continuing this path by doing their own research.

Bibliography

Abdulle, A. and Garegnani, G. (2021). A probabilistic finite element method based on random meshes: A posteriori error estimators and Bayesian inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 384:113961.

Alvarez, M., Luengo, D., and Lawrence, N. D. (2009). Latent force models. In *AISTATS 2009*.

Ambrosio, B. and Françoise, J.-P. (2009). Propagation of bursting oscillations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1908):4863–4875.

Axelsson, P. and Gustafsson, F. (2015). Discrete-time solutions to the continuous-time differential Lyapunov equation with applications to Kalman filtering. *IEEE Transactions on Automatic Control*, 60(3):632–643.

Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. (2004). *Estimation With Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons.

Bell, B. M. (1994). The iterated Kalman smoother as a Gauss–Newton method. *SIAM Journal on Optimization*, 4(3):626–636.

Bentley, J. L. and Ottmann, T. A. (1979). Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, 28(09):643–647.

Berzins, M. (1988). Global error estimation in the method of lines for parabolic equations. *SIAM Journal on Scientific and Statistical Computing*, 9(4):687–703.

Berzins, M., Baehmann, P., Flaherty, J., and Lawson, J. (1991). Towards an automated finite element solver for time-dependent fluid-flow problems. *The Mathematics of Finite Elements and Application*, 7:181–188.

Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv:1701.02434*.

Bosch, N., Hennig, P., and Tronarp, F. (2021). Calibrated adaptive probabilistic ODE solvers. In *AISTATS 2021*.

Calderhead, B., Girolami, M., and Lawrence, N. (2009). Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes. In *NeurIPS 2009*.

Cash, J. and Psihoyios, Y. (1996). The MOL solution of time dependent partial differential equations. *Computers & Mathematics with Applications*, 31(11):69–78.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *NeurIPS 2018*.

Chen, Y., Hosseini, B., Owhadi, H., and Stuart, A. M. (2021). Solving and learning nonlinear PDEs with Gaussian processes. *Journal of Computational Physics*, 447.

Chkrebtii, O. A., Campbell, D. A., Calderhead, B., and Girolami, M. A. (2016). Bayesian solution uncertainty quantification for differential equations. *Bayesian Analysis*, 11(4):1239–1267.

Cockayne, J., Oates, C., Sullivan, T., and Girolami, M. (2017). Probabilistic numerical methods for PDE-constrained Bayesian inverse problems. In *AIP Conference Proceedings*, volume 1853.

Cockayne, J., Oates, C. J., Sullivan, T. J., and Girolami, M. (2019). Bayesian probabilistic numerical methods. *SIAM Review*, 61(4):756–789.

Conrad, P. R., Girolami, M., Särkkä, S., Stuart, A., and Zygalakis, K. (2017). Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Statistics and Computing*, 27(4):1065–1082.

Dereli, Y. and Schaback, R. (2013). The meshless kernel-based method of lines for solving the equal width equation. *Applied Mathematics and Computation*, 219(10):5224–5232.

Dong, E., Du, H., and Gardner, L. (2020). An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, 20(5):533–534.

Dormand, J. and Prince, P. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

Driscoll, T. A. and Fornberg, B. (2002). Interpolation in the limit of increasingly flat radial basis functions. *Computers & Mathematics with Applications*, 43(3-5):413–422.

Duffin, C., Cripps, E., Stemler, T., and Girolami, M. (2021). Statistical finite elements for misspecified models. *Proceedings of the National Academy of Sciences*, 118(2).

Fasshauer, G. E. (1997). Solving partial differential equations by collocation with radial basis functions. In *Surface Fitting and Multiresolution Methods*, pages 131–138. University Press.

Fasshauer, G. E. (1999). Solving differential equations with radial basis functions: multilevel methods and smoothing. *Advances in Computational Mathematics*, 11(2):139–159.

Fasshauer, G. E. (2007). *Meshfree Approximation Methods with MATLAB*. World Scientific.

Fornberg, B. (1988). Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699–706.

Fornberg, B. and Flyer, N. (2015). *A Primer on Radial Basis Functions with Applications to the Geosciences*. SIAM.

Frank, P. and Enßlin, T. A. (2020). Probabilistic simulation of partial differential equations. *arXiv preprint arXiv:2010.06583*.

Gai, C., Iron, D., and Kolokolnikov, T. (2020). Localized outbreaks in an S-I-R model with diffusion. *Journal of Mathematical Biology*, 80:1389–1411.

Gear, C. W. (1980). Runge–Kutta starters for multistep methods. *ACM Transactions on Mathematical Software (TOMS)*, 6(3):263–279.

Gelbrecht, M., Boers, N., and Kurths, J. (2021). Neural partial differential equations for chaotic systems. *New Journal of Physics*, 23(4):043005.

Giordano, G., Blanchini, F., Bruno, R., Colaneri, P., Di Filippo, A., Di Matteo, A., and Colaneri, M. (2020). Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy. *Nature Medicine*, pages 1–6.

Girolami, M., Febrianto, E., Yin, G., and Cirak, F. (2021). The statistical finite element method (statFEM) for coherent synthesis of observation data and model predictions. *Computer Methods in Applied Mechanics and Engineering*, 375.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *ICLR 2018*.

Grewal, M. S. and Andrews, A. P. (2014). *Kalman Filtering: Theory and Practice with MATLAB*. John Wiley & Sons.

Griewank, A. and Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM.

Guckenheimer, J. (1980). Dynamics of the van der Pol equation. *IEEE Transactions on Circuits and Systems*, 27(11):983–989.

Gustafsson, K. (1992). *Control of Error and Convergence in ODE solvers*. PhD thesis, Lund University.

Hairer, E., Nørsett, S. P., and Wanner, G. (1993). *Solving Ordinary Differential Equations I, Nonstiff Problems*. Springer.

Hartikainen, J., Seppänen, M., and Särkkä, S. (2012). State-space inference for non-linear latent force models with application to satellite orbit prediction. *ICML 2012*.

Hennig, P. and Hauberg, S. (2014). Probabilistic solutions to differential equations and their application to Riemannian statistics. In *AISTATS 2014*.

Hennig, P., Osborne, M. A., and Girolami, M. (2015). Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179).

Hethcote, H. W. (2000). The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653.

Hindmarsh, A. and Petzold, L. (2005). LSODA, ordinary differential equation solver for stiff or non-stiff system.

Hoffman, M. D. and Gelman, A. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623.

Holmes, E., Lewis, M., Banks, J., and Veit, D. (1994). Partial differential equations in ecology: Spatial interactions and population dynamics. *Ecology*, 75:17–29.

Hon, Y. and Schaback, R. (2001). On unsymmetric collocation by radial basis functions. *Applied Mathematics and Computation*, 119(2-3):177–186.

Hon, Y., Schaback, R., and Zhong, M. (2014). The meshless kernel-based method of lines for parabolic equations. *Computers & Mathematics with Applications*, 68(12):2057–2067.

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory*. Academic Press.

Jidling, C., Wahlström, N., Wills, A., and Schön, T. B. (2017). Linearly constrained Gaussian processes. In *NeurIPS 2017*.

John, D., Heuveline, V., and Schober, M. (2019). GOODE: A Gaussian off-the-shelf ordinary differential equation solver. In *ICML 2019*.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Kansa, E. J. (1990). Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—II solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & Mathematics with Applications*, 19(8-9):147–161.

Kersting, H. and Hennig, P. (2016). Active uncertainty calibration in Bayesian ODE solvers. *UAI 2016*.

Kersting, H., Krämer, N., Schiegg, M., Daniel, C., Tiemann, M., and Hennig, P. (2020a). Differentiable likelihoods for fast inversion of 'likelihood-free' dynamical systems. In *ICML 2020*.

Kersting, H., Sullivan, T. J., and Hennig, P. (2020b). Convergence rates of Gaussian ODE filters. *Statistics and Computing*, 30(6):1791–1816.

Krämer, N., Bosch, N., Schmidt, J., and Hennig, P. (2021a). Probabilistic ODE solutions in millions of dimensions. *arXiv preprint arXiv:2110.11812*.

Krämer, N. and Hennig, P. (2020). Stable implementation of probabilistic ODE solvers. *arXiv:2012.10106*.

Krämer, N. and Hennig, P. (2021). Linear-time probabilistic solutions of boundary value problems. *arXiv preprint arXiv:2106.07761*.

Krämer, N., Schmidt, J., and Hennig, P. (2021b). Probabilistic numerical method of lines for time-dependent partial differential equations.

Lawson, J., Berzins, M., and Dew, P. M. (1991). Balancing space and time errors in the method of lines for parabolic equations. *SIAM Journal on Scientific and Statistical Computing*, 12(3):573–594.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Fourier neural operator for parametric partial differential equations. In *ICLR 2021*.

Lindgren, F., Rue, H., and Lindström, J. (2011). An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528.

Loper, J., Blei, D. M., Cunningham, J. P., and Paninski, L. (2020). General linear-time inference for Gaussian processes on one dimension. *arXiv:2003.05554*.

Lorenz, E. N. (1996). Predictability: A problem partly solved. In *Proceedings of the Seminar on Predictability*, volume 1.

Lotka, A. J. (1978). The growth of mixed populations: two species competing for a common food supply. In *The Golden Age of Theoretical Ecology: 1923–1940*, pages 274–286. Springer.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229.

Maybeck, P. S. (1982). *Stochastic Models, Estimation, and Control*. Academic Press.

Murray, I., Adams, R., and MacKay, D. (2010). Elliptical slice sampling. In *AISTATS 2010*.

Naesseth, C. A., Lindsten, F., and Schön, T. B. (2019). Elements of sequential Monte Carlo. *Foundations and Trends® in Machine Learning*, 12(3):307–392.

Øksendal, B. (2003). *Stochastic Differential Equations*. Springer.

Owhadi, H. (2015). Bayesian numerical homogenization. *Multiscale Modeling & Simulation*, 13(3):812–828.

Owhadi, H. (2017). Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games. *SIAM Review*, 59(1):99–149.

Phan, D., Pradhan, N., and Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv:1912.11554*.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959.

Rackauckas, C. and Nie, Q. (2017). DifferentialEquations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5(1).

Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *NeurIPS 2008*.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, 348:683–693.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2018). Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press.

Särkkä, S. and Solin, A. (2019). *Applied Stochastic Differential Equations*. Cambridge University Press.

Schaback, R. (1995). Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(3):251–264.

Schaback, R. (2007). Convergence of unsymmetric kernel-based meshless collocation methods. *SIAM Journal on Numerical Analysis*, 45(1):333–351.

Schiesser, W. E. (2012). *The Numerical Method of Lines: Integration of Partial Differential Equations*. Elsevier.

Schmidt, J., Kraemer, N., and Hennig, P. (2021). A probabilistic state space model for joint inference from differential equations and data. In *NeurIPS 2021*.

Schober, M., Duvenaud, D. K., and Hennig, P. (2014). Probabilistic ODE solvers with Runge–Kutta means. *NeurIPS 2014*.

Schober, M., Särkkä, S., and Hennig, P. (2019a). A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122.

Schober, M., Särkkä, S., and Hennig, P. (2019b). A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29:99–122.

Schweppe, F. (1965). Evaluation of likelihood functions for Gaussian signals. *IEEE Transactions on Information Theory*, 11(1):61–70.

Shampine, L. F. and Reichelt, M. W. (1997). The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22.

Shu, C., Ding, H., and Yeo, K. (2003). Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier–Stokes equations. *Computer methods in applied mechanics and engineering*, 192(7-8):941–954.

Sidén, P. and Lindsten, F. (2020). Deep Gaussian Markov random fields. In *ICML 2020*.

Solin, A. (2016). *Stochastic Differential Equation Methods for Spatio-Temporal Gaussian Process Regression*. Doctoral thesis, School of Science.

Solin, A. and Särkkä, S. (2014). Explicit link between periodic covariance functions and state space models. In *AISTATS 2014*.

Stengel, R. (1994). *Optimal Control and Estimation*. Dover Publications.

Tolstykh, A. and Shirobokov, D. (2003). On using radial basis functions in a "finite difference mode" with applications to elasticity problems. *Computational Mechanics*, 33(1):68–79.

Tronarp, F., García-Fernández, Á. F., and Särkkä, S. (2018). Iterative filtering and smoothing in nonlinear and non-Gaussian systems using conditional moments. *IEEE Signal Processing Letters*, 25(3):408–412.

Tronarp, F., Kersting, H., Särkkä, S., and Hennig, P. (2019). Probabilistic solutions to ordinary differential equations as non-linear Bayesian filtering: A new perspective. *Statistics and Computing*, 29(6):1297–1315.

Tronarp, F., Särkkä, S., and Hennig, P. (2021). Bayesian ODE solvers: The maximum a posteriori estimate. *Statistics and Computing*, 31(3):1–18.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272.

Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158.

Wang, J., Cockayne, J., Chkrebtii, O., Sullivan, T. J., and Oates, C. J. (2021). Bayesian numerical methods for nonlinear partial differential equations. *Statistics and Computing*, 31.

Wanner, G. and Hairer, E. (1996). *Solving Ordinary Differential Equations II, Stiff Problems*, volume 375. Springer.

Wenger, J., Krämer, N., Pförtner, M., Schmidt, J., Bosch, N., Effenberger, N., Zenn, J., Gessner, A., Karvonen, T., Briol, F.-X., Mahsereci, M., and Hennig, P. (2021). Probnum: Probabilistic numerics in python.

Wenk, P., Abbati, G., Osborne, M. A., Schölkopf, B., Krause, A., and Bauer, S. (2020). ODIN: ODE-informed regression for parameter and state inference in time-continuous dynamical systems. *AAAI 2020*, 34(04):6364–6371.

A Sources for governmental measures in Germany

This section provides the sources used to list the governmental measures in Table 3.1. In order to provide reliable sources, we refer to the official press releases, as published by the German government. For each policy change, we provide a very brief idea of the imposed measures and official sources by the German government (only available in German language).

March 22, 2020 (Mark 1)

Citizens are urged to restrict social contacts as much as possible and the formation of groups is sanctioned in public spaces as well as at home.

https://www.bundesregierung.de/breg-de/themen/coronavirus/besprechung-der-bundeskanzlerin-mit-den-regierungschefinnen-und-regierungschefs-der-laender-vom-22-03-2020-1733248

https://www.bundesregierung.de/resource/blob/975226/1733246/e6d6ae0e89a7ffea1ebf6f32cf472736/2020-03-22-mpk-data.pdf?download=1

May 6, 2020 (Mark 2)

The government puts the federal states in charge of appropriately relaxing the imposed measures. Different states handle the situation differently, according to the respective incidences (*'hotspot strategy'*).

https://www.bundesregierung.de/breg-de/aktuelles/pressekonferenzen/pressekonferenz-von-bundeskanzlerin-merkel-ministerpraesident-soeder-und-dem-ersten-buergermeister-tschentscher-im-anschluss-an-das-gespraech-mit-den-regierungschefinnen-und-regierungschefs-der-laender-1751050

October 7, 2020 (Mark 3) and October 14, 2020

The population is again urged to restrict contacts if possible.

https://www.bundeskanzlerin.de/bkin-de/aktuelles/telefonschaltkonferenz-des-chefs-des-bundeskanzleramts-mit-den-chefinnen-und-chefs-der-staats-und-senatskanzleien-der-laender-am-7-oktober-2020-1796770

One week later, new light restrictions are imposed. The number of people allowed in social gatherings is limited, according to local incidences.

https://www.bundesregierung.de/resource/blob/997532/1798920/9448da53f1fa442c24c37abc8b0b2048/2020-10-14-beschluss-mpk-data.pdf?download=1

November 2, 2020 (Mark 4)

Partial shutdown of public life (*'lockdown light'*). Across the country, the number of people allowed in social gatherings is limited to ten, where the number of households present must not exceed two. Most of public services are closed or offered only virtually, if possible.

https://www.bundesregierung.de/breg-de/aktuelles/videokonferenz-der-bundeskanzlerin-mit-den-regierungschefinnen-und-regierungschefs-der-laender-am-28-oktober-2020-1805248

December 16, 2020 (Mark 5)

Across the country, the number of people allowed in social gatherings is limited to five, where the number of households present must not exceed two. Except for stores of systemic importance, the retail sector is mostly shut down.

https://www.bundesregierung.de/resource/blob/997532/1827366/69441fb68435a7199b3d3a89bff2c0e6/2020-12-13-beschluss-mpk-data.pdf?download=1

April 23, 2021 (Mark 6)

The aforementioned measures were mostly governed and implemented by the respective federal states. On April 22, 2021, the German government decides on a nationwide decree of measures to come into effect on the following day (April 23, 2021). Depending on the seven-day incidence, curfews, contact restrictions, and a shutdown of large parts of public life are imposed.

https://www.bundesgesundheitsministerium.de/fileadmin/Dateien/3_Downloads/Gesetze_und_Verordnungen/GuV/B/4_BevSchG_BGBL.pdf

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum	Unterschrift (Jonathan Schmidt)